Tugas Akhir – TE141599

PENGATURAN KECEPATAN MOTOR *SPINDLE* SAAT PROSES *FACE MILLING* PADA CNC MENGGUNAKAN KONTROLER *NEURO-PID*

Uci Adriati
NRP 2213106079

Dosen Pembimbing
Ir. Joko Susila, MT.
Ir. Rusdhianto Effendie A.K, MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

Final Project – TE141599

SPINDLE MOTOR SPEED CONTROL OF FACE MILLING PROCESS ON CNC USING NEURO-PID CONTROLLER

Uci Adriati
NRP 2213106079

Advisor
Ir. Joko Susila, MT.
Ir. Rusdhianto Effendie A.K, MT.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institut of Technology
Surabaya 2016

PENGATURAN KECEPATAN MOTOR *SPINDLE* SAAT PROSES *FACE MILLING* PADA CNC MENGGUNAKAN KONTROLER *NEURO-PID*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Joko Susila, MT.	Ir. Rusdhianto Effendie A.K, MT.
NIP. 1966 06 06 1991 02 1001	NIP. 1957 04 24 1985 02 1001

SURABAYA
JANUARI, 2016

PENGATURAN KECEPATAN MOTOR *SPINDLE* SAAT PROSES *FACE MILLING* PADA CNC MENGGUNAKAN KONTROLER *NEURO-PID*

Nama : Uci Adriati
Pembimbing I : Ir. Joko Susila, MT.
Pembimbing II : Ir. Rusdhianto Effendie A.K, MT.

ABSTRAK

Mesin CNC adalah suatu mesin yang dikontrol oleh komputer dengan menggunakan bahasa numerik berupa data perintah dengan kode angka, huruf dan simbol sesuai standar ISO. Mesin CNC yang digunakan adalah jenis *milling*. Salah satu permasalahan pada CNC *milling* adalah pemberian beban yang berubah-ubah akan mempengaruhi kinerja motor *spindle* saat proses *face milling*. *Face milling* itu sendiri adalah proses saat mata pahat tegak lurus dengan permukaan benda kerja untuk melakukan proses *feeding* (pemakanan). Perubahan beban menyebabkan perubahan respon sistem. Perubahan beban diindikasikan dengan adanya perubahan arus yang mengalir pada motor *spindle*. Untuk menjaga kestabilan kecepatan motor *spindle* pada kondisi perubahan beban tertentu, diperlukan sebuah desain kontroler. Kontroler yang digunakan adalah *Neuro-PID*. *Neuro-PID* adalah kontroler Jaringan Syaraf Tiruan (*Artificial Neural Network*) yang digunakan untuk tuning parameter PID sesuai kondisi beban yang berubah. Nilai K_p, K_i dan K_d pada masing-masing kondisi beban minimal, nominal, dan maksimal yaitu Kp_1=0,5254; Ki_1=2,466; Kd_1=0,0249; Kp_2=1,0375; Ki_2=2,4655; Kd_2=0,0989; Kp_3=4,075; Ki_3=4,9006; Kd_3=0,7101. Pada pengujian simulasi, kontroler *Neuro-PID* mampu mengontrol kestabilan kecepatan motor *spindle* dengan kecepatan referensi yang berubah dari 2000 rpm sampai 3000 rpm dan kondisi beban dengan perubahan tertentu. Hal ini ditunjukkan dengan kecilnya nilai *error overshoot* yang hanya terjadi saat perubahan beban minimal kemaksimal di kecepatan 3000 rpm yaitu sebesar 0,23%.

Kata Kunci: CNC, Motor *Spindle*, *Neuro-PID*

Halaman ini sengaja dikosongkan

SPINDLE MOTOR SPEED CONTROL OF FACE MILLING PROCESS ON CNC USING NEURO-PID CONTROLLER

Name : **Uci Adriati**
Advisor I : **Ir. Joko Susila, MT.**
Advisor II : **Ir. Rusdhianto Effendie A.K, MT.**

ABSTRACT

CNC machine is a machine that is controlled by a computer using numerical language in the form of command data with the code numbers, letters and symbols corresponding ISO standard. CNC machine used is the type of milling. One of the problems in CNC milling is the provision of load change will affect the performance of the spindle motor in face milling process. Face milling is a process when the eyes chisel perpendicular to the surface of the workpiece to perform the process of feeding. Load changes cause changes in system response. Load change is indicated by a change in current flowing to the motor spindle. To maintain the stability of the spindle motor speed at certain load conditions change, we need a controller design. The controller used is Neuro-PID. Neuro-PID is a Artificial Neural Network controller which is used for the appropriate PID tuning parameters changing load conditions. The value of Kp, Ki and Kd for each load condition minimal, nominal and maximal namely $Kp_1 = 0{,}5254$; $Ki_1 = 2{,}466$; $Kd_1 = 0{,}0249$; $Kp_2 = 1{,}0375$; $Ki_2 = 2{,}4655$; $Kd_2 = 0{,}0989$; $Kp_3 = 4{,}075$; $Ki_3 = 4{,}9006$; $Kd_3 = 0{,}7101$. *In the simulation test, Neuro-PID controller is able to control the stability of the speed of the spindle motor with a speed reference changes from 2000 rpm to 3000 rpm and load conditions with certain changes. This is indicated by the small value overshoot error only occurs when the load changes minimal to maximal load at 3000 rpm which is at 0,23%.*

Keywords : CNC, *Spindle Motor*, *Neuro-PID*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik. Tak lupa Shalawat serta salam selalu tercurah kepada junjungan besar Nabi Muhammad SAW beserta keluarga, sahabat dan para pengikutnya.

Penulis ingin mengucapkan terimakasih kepada beberapa pihak yang telah berperan dalam mendukung pengerjaan Tugas Akhir ini:

1. Ayahanda Ir. Suripto, M.Eng, Ibunda Sri Swidati, kakak dan adik yang telah setia dalam doa, usaha dan materi demi mendukung penulis untuk menyelesaikan pendidikan S1 ini dengan baik.
2. Bapak Ir. Joko Susila, MT, selaku dosen pembimbing 1 dan Bapak Ir. Rusdhianto Effendie A.K, MT, selaku dosen pembimbing 2 yang telah membantu dan membimbing penulis dengan sabar sehingga Tugas Akhir ini dapat terselesaikan dengan baik.
3. Bapak Dr. Eng. Ardyono Priyadi, ST, M. Eng, selaku Ketua Jurusan Teknik Elektro – ITS.
4. Seluruh dosen Jurusan Teknik Elektro-ITS, yang telah banyak memberikan ilmu serta motivasi dalam memahami setiap ilmu yang dipelajari.
5. Teman-teman seperjuangan TA, Lintas Jalur angkatan 2013 Genap yang telah menghiasi hari-hari penulis dalam suka dan duka.
6. Para Aslab B105 yang turut membantu penulis untuk memperoleh referensi dan fasilitas penunjang dalam pengerjaan Tugas Akhir.
7. Rina Wulandari Subagyo, terimakasih atas persahabatan dan dukungan yang telah diberikan selama 20 tahun terakhir ini.

Penulis menyadari bahwa masih terdapat kekurangan di dalam Tugas Akhir ini maka kritik dan saran sangat penulis harapkan untuk perbaikan di masa yang akan datang. Akhir kata penulis ucapkan terimakasih dan semoga Tugas Akhir ini dapat memberi manfaat bagi para pembaca.

Surabaya, Januari 2016

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Mesin CNC (*Computer Numerical Control*) merupakan suatu mesin yang dikontrol oleh komputer dengan menggunakan bahasa numerik berupa data perintah dengan kode angka, huruf dan simbol yang sesuai dengan standar ISO. Komputer tersebut memberi intruksi agar CNC bekerja sesuai dengan program yang dibuat sehingga dapat menghasilkan pola pada benda kerja sesuai dengan yang diinginkan. Pada CNC jenis *milling*, terdapat tiga tahapan proses yaitu *slab milling*, *face milling* dan *end milling*. *Slab milling* adalah proses ketika mata pahat sejajar dengan benda kerja untuk menghaluskan permukaannya. *Face milling* adalah proses ketika mata pahat tegak lurus dengan permukaan benda kerja. *End milling* adalah proses ketika mata pahat yang berbentuk lurus, bergerak pada sumbu tegak lurus untuk proses penghalusan terakhir.

Pada proses *face milling*, tiga motor penggerak sumbu akan bergerak secara bersamaan dengan kecepatan tertentu untuk menggerakkan benda kerja dan sebuah motor untuk menggerakkan *spindle*. *Spindle* adalah bagian mesin CNC yang berputar secara cepat dengan kecepatan tertentu yang digunakan untuk membentuk pola pada benda kerja.

Pada penelitian ini, mesin CNC diharapkan dapat beradaptasi terhadap perubahan beban tertentu. Metode pembebanan yang dilakukan adalah dengan memberikan kecepatan gerak potong (*feedrate*) yang berbeda-beda pada setiap pola benda kerja. Perubahan kecepatan gerak potong tersebut diperlukan untuk mengetahui respon sistem ketika mesin CNC membentuk pola pada benda kerja dengan kedalaman tertentu.

CNC merupakan sistem lup terbuka yang tidak dapat memperbaiki *error*-nya sendiri jika terjadi perubahan kecepatan referensi akibat perubahan beban yang muncul. Untuk mengatasi hal tersebut diperlukanlah sebuah kontroler yang dapat mengatur kecepatan motor *spindle* tetap stabil dalam referensi walau terjadi perubahan beban, agar pada saat *face milling* pola yang dibuat pada benda kerja sesuai harapan. Kontroler yang dipilih pada penelitian ini adalah *Neuro-PID*. Kontroler PID dipilih karena perhitungannya yang sederhana dan mudah diimplementasikan. Namun kontroler PID saja tidak cukup untuk mengontrol kondisi beban yang berubah-ubah. Maka dibutuhkan

Artificial Neural Network atau Jaringan Syaraf Tiruan (JST) sebagai tuning untuk PID agar dapat beradaptasi dengan kondisi beban yang berubah-ubah. *Neural Network* ini digunakan untuk menentukan nilai parameter Kp, Ki, dan Kd yang sesuai dengan kondisi beban minimal, nominal dan maksimal. Sehingga dapat menghasilkan respon transien sistem yang diinginkan.

1.2 Perumusan Masalah

Saat terjadi proses *face milling* pada CNC, diperlukan adanya penyesuaian kecepatan motor *spindle* dari yang awalnya tidak berbeban menjadi berbeban. Respon kecepatan motor *spindle* saat proses *feeding* harus dapat dijaga konstan mengikuti referensi walaupun terjadi perubahan beban. Perubahan beban dapat diketahui dari perubahan arus. Perubahan arus yang terlalu besar dapat menyebabkan kecepatan gerak motor *spindle* yang berlebihan. Hal ini dapat membahayakan mata pahat. Selain itu, tidak adanya mekanisme kontrol pada CNC juga menyebabkan mesin tetap bekerja walaupun diberi beban berlebih.

Oleh karena itu diperlukan suatu desain kontroler yang mampu menagatur kecepatan motor *spindle* selama proses *face milling* saat beban berubah agar tetap memenuhi spesifikasi respon sistem yang diinginkan.

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian ini adalah sebagai berikut:

1. Pemodelan sistem yang dilakukan adalah dengan metode identifikasi statis.
2. Pembebanan *plant* adalah dengan memberi *feedrate* yang berbeda saat *feeding* untuk menghasilkan pola segi empat pada benda kerja jenis plastik polietilen.
3. Simulasi pada *plant* dilakukan dengan memberi kecepatan referensi antara 2000 rpm dan 3000 rpm yang berubah setiap 40 detik dan kondisi perubahan beban tertentu setiap 10 detik.
4. Proses *learning Neural Network* dilakukan secara *offline*.
5. Respon transien sistem yang diinginkan yaitu mendekati fungsi alih sistem orde 1.

1.4 Tujuan

Penelitian ini bertujuan untuk mengatur kecepatan motor *spindle* pada proses *face milling* agar dapat beradaptasi memenuhi spesifikasi respon transien tertentu saat diberi beban yang berubah-ubah dengan merancang kontroler *Neuro – PID*.

1.5 Metodologi

Metodologi yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut:

a. Studi Literatur
   Dilakukan untuk memperoleh informasi mengenai CNC, motor *spindle*, kontroler PID, metode Jaringan Syaraf Tiruan dan *Neuro-PID* melalui buku teks, jurnal, artikel, internet dan lain-lain.

b. Pemodelan Sistem
   Hasil yang didapat dari studi literatur disesuaikan dengan keadaan *plant* sesungguhnya. Dengan demikian, pada tahap ini akan dilakukan pengambilan data dari mesin CNC. Setelah mendapatkan data yang dibutuhkan, akan dilakukan identifikasi dari *plant* untuk mendapatkan model matematika yang akan digunakan untuk mendesain kontroler sesuai dengan yang diharapkan.

c. Perancangan Kontroler
   Pada tahap ini dibuat struktur kontroler *Neuro-PID* untuk pengaturan kecepatan motor *spindle* yang merupakan jenis motor DC.

d. Pengujian Simulasi
   Hasil pemodelan sistem dan desain kontroler disimulasikan dengan menggunakan perangkat lunak Matlab.

e. Penulisan Buku Tugas Akhir
   Buku tugas akhir ditulis secara intensif bila proses pengujian telah selesai. Pada saat proses pengujian sedang berjalan, dilakukan eksplorasi bahan-bahan untuk penulisan buku Tugas Akhir dari jurnal-jurnal ilmiah dan buku. Penulisan buku tugas akhir meliputi pendahuluan, teori dasar, perancangan sistem, simulasi dan analisa, serta penutup.

1.6 Sistematika Penulisan

Pada Tugas Akhir ini sistematika penulisan dibagi menjadi lima bab, yaitu:

Bab 1 : Pendahuluan

Menjelaskan tentang latar belakang pemilihan topic, perumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan dan relevansi.

Bab 2 : Teori Penunjang

Menjelaskan tentang dasar teori yang menjadi penunjang dalam pengerjaan Tugas Akhir. Teori dasar meliputi CNC, metode identifikasi sistem, kontroler PID, *Neural Network* dan konsep kontroler *Neuro*-PID).

Bab 3 : Perancangan Sistem

Menjelaskan tentang identifikasi sistem dari motor *spindle* CNC dan perancangan kontroler *Neuro*-PID pada *software* Matlab.

Bab 4 : Pengujian dan Analisa

Menjelaskan tetang cara pengujian sistem dan menganalisa dari respon hasil simulasi sistem.

Bab 5 : Penutup

Berisi kesimpulan dan saran dari perancangan kontroler *Neuro-PID* yang telah dibuat untuk pengaturan kecepatan motor *spindle* saat proses *face milling* pada CNC.

1.7 Relevansi

Hasil perancangan kontroler *Neuro-PID* pada motor *spindle* diharapkan dapat memberikan gambaran bagaimana kontroler tersebut bekerja serta pengaruhnya terhadap *plant* dalam kondisi perubahan beban tertentu selama proses *face milling*. Untuk penelitian selanjutnya Tugas Akhir ini dapat dijadikan sebagai acuan agar dapat dirancang kontroler yang lebih baik lagi yang dapat tahan terhadap perubahan beban yang lebih bervariasi.

BAB II
TEORI PENUNJANG

Bab 3 membahas tentang teori penunjang yang dijadikan acuan dalam melakukan penelitian. Secara kesuluruhan, bab ini akan membahas mesin CNC, identifikasi sistem, kontroler PID, Jaringan Syaraf Tiruan dan kontroler Neuro-PID.

2.1 Mesin CNC

Mesin CNC adalah suatu mesin yang dikontrol oleh computer dengan menggunakan bahasa numeric berupa data perintah dengan kode angka, huruf dan symbol sesuai standar ISO. Mesin CNC mampu melakukan operasi yang bervariasi dalam membentuk kerja, antara lain membentuk permukaan datar, muka bersudut, alur atau untuk pembentukan lubang dengan bentuk tertentu (*pocketing*). Mesin CNC memiliki dua atau lebih arah gerakan *tool* yang disebut dengan sumbu atau *axis*. Gerakan pada *axis* antara lain linear (yang merupakan garis lurus) atau gerakan *circular* (yang merupakan lintasan melingkar). Umumnya, sumbu yang terdapat pada gerakan linear adalah X, Y dan Z sedangkan nama *axis* pada gerakan *circular* adalah A, B dan C [1].

Komputer sebagai perangkat masukan mesin CNC sangat penting peranannya untuk mengatur kinerjanya. Komputer tersebut memberi intruksi ke mesin CNC agar bekerja sesuai dengan program yang dibuat, sehingga dapat menghasilkan pola dan bentuk material sesuai dengan yang diinginkan. Mesin CNC hanya dapat membaca kode standar yang telah disepakati oleh industri pembuat mesin CNC. Dengan kode standar tersebut, pabrik mesin CNC dapat menggunakan computer yang diproduksi sendiri atau yang direkomendasikan. Kode standar pada mesin CNC dikenal dengan nama *G-code*. Selain itu, pemograman mesin CNC juga dapat dilakukan dengn *G-code* nonstandar yang hanya terdapat pada mesin CNC tertentu [2].

Salah satu spesifikasi yang dapat memperlihatkan kekompleksitasan dari mesin CNC adalah berapa banyak *axis* yang dimilikinya dan gerakan interpolasi yang ada. Prototipe mesin CNC yang digunakan pada Tugas Akhir ini adalah mesin CNC berjenis mesin CNC *Milling* dengan tipe *Complete CNC Mill Package* 35-005-C. Mesin *milling* adalah sebuah jenis mesin CNC yang dapat mencetak pola tertentu pada suatu permukaan benda kerja dimana dapat diprogram

pada tiga sumbu, yaitu sumbu X untuk mengontrol gerakan ke kiri dan ke kanan, sumbu Y untuk mengontrol gerakan menuju atau jauh dari kolom, dan sumbu Z untuk mengontrol gerakan vertikal (atas atau bawah) dari *spindle* (lengan). Gambar 2.1 menunjukkan prototipe mesin CNC yang digunakan pada pengerjaan Tugas Akhir ini.

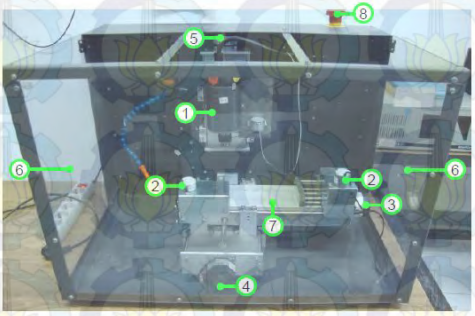
Gambar 2.1 Mesin CNC *Mill Package 35-005-C*

2.1.1 Bagian Utama CNC *Milling*

Seperti yang ditunjukkan pada Gambar 2.2, mesin CNC memiliki bagian utama sebagai penunjang untuk melakukan proses perautan pada benda kerja. Umumnya bagian mesin CNC *milling* terdiri dari meja kerja, *spindle*, *chuck*, *vice*, *coolant hose,* panel kontrol, monitor. Meja kerja digunakan untuk meletakkan benda kerja yang akan diraut. Mesin CNC *milling* bisa bergerak dalam 2 sumbu yaitu sumbu X dan Y. Untuk masing-masing sumbunya, meja kerja dilengkapi dengan motor penggerak, *ball screw plus bearing* dan *guide way slider* untuk akurasi pergerakannya. Untuk pelumasannya, beberapa CNC menggunakan minyak oli dengan jenis dan merk tertentu. Beberapa mesin menggunakan *grease*. Pelumasan ini sangat penting untuk menjaga kehalusan pergerakan meja sehingga dapat menghindari kerusakan pada *ball screw*, *bearing* dan *guide slider*. Untuk itu pemberian pelumas setiap pemakaian CNC wajib dilakukan. Meja kerja ini dapat digerakkan secara manual dengan menggunakan *handle* eretan dan digerakkan secara otomatis yang dapat bekerja sesuai perintah dari koordinat yang ditentukan oleh program G-code. dengan motor penggerak sumbu.

Spindle adalah bagian dari mesin yang menjadi rumah cutter. *Spindle* inilah yang mengatur putaran dan pergerakkan mata bor. S*pindle*

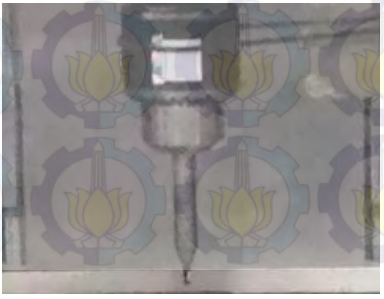
digerakkann oleh motor yang dilengkapi oleh transmisi berupa *belting* atau *kopling*. *Chuck* adalah alat cekam untuk meletakkan mata bor. Bagian ini dapat dibongkar pasang dan diganti dengan mata bor yang sesuai dengan bentuk benda kerja yang diinginkan. *Vice* adalah bagian mesin CNC yang digunakan untuk mencekam benda kerja pada meja kerja agar tidak bergerak selama proses *feeding*/pemakanan benda kerja berlangsung.



Gambar 2.2 Bagian Utama CNC *Milling* [2]

Keterangan nomor pada Gambar 2.2 yaitu:

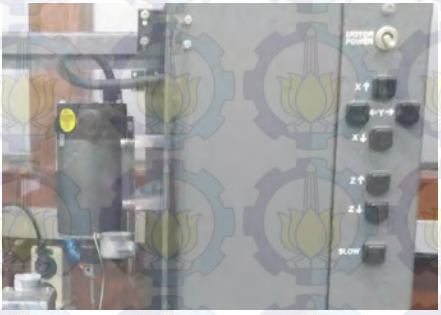
1. Motor DC *spindle*
2. Motor DC *vice*
3. Motor *stepper* penggerak sumbu X
4. Motor *stepper* penggerak sumbu Y
5. Motor *stepper* penggerak sumbu Z
6. Pintu yang berfungsi seperti sendor
7. Meja kerja
8. Tombol *emergency*



Gambar 2.3 *Chuck*

Gambar 2.4 *Coolant Hose*

Coolant hose digunakan untuk pendingin mata bor dan benda kerja. Hal ini diperlukan agar pada saat proses feeding, CNC dalam keadaan suhu yang baik.



Gambar 2.5 Tombol Sumbu X, Y dan Z

2.1.2 Prinsip Kerja [2]

Seperti gambar mesin CNC *milling* yang ditunjukkan pada Gambar 2.1 dan Gambar 2.2, dapat diilustrasikan bagaimana sebuah mesin CNC *milling* bekerja. Benda kerja yang akan dipahat, dipasang pada meja kerja. Kemudian, program *G-code* yang berasal dari komputer atau media pemrogaman lainnya dijalankan. Rangkaian elektrik mesin akan menggerakkan motor-motor selama proses *feeding* berlangsung. *Vice* akan bergerak untuk mengeratkan benda kerja dan motor *spindle* akan bergerak untuk mengambil posisi awal pada koordinat tertentu sebelum melakukan proses pengeboran. Motor *spindle* bergerak selama proses *feeding* dan motor penggerak sumbu akan bergerak sesuai dengan koordinat yang diberikan pada program *G-*

code. Motor *spindle* akan terus bergerak membentuk pola pada benda kerja sesuai dengan yang diinginkan sampai proses selesai. Untuk mesin CNC yang digerakkan secara manual, sumbu- sumbu tersebut dapat digerakkan dengan tombol-tombol atau sejenisnya dan pengaturannya disesuaikan dengan keinginan operator seperti yang ditunjukkan pada Gambar 2.5.

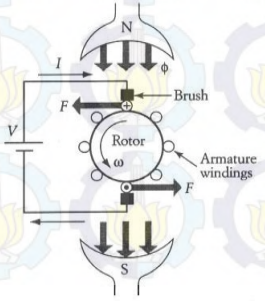
Pada suatu mesin CNC *milling*, umumnya terdapat 3 buah sumbu koordinat yaitu sumbu X, Y, dan Z. Sumbu X dan Y berfungsi untuk membentuk pola secara horizontal, sedangkan sumbu Z digunakan untuk membentuk pola secara vertikal atau melakukan proses pengeboran.

2.1.3 Motor DC Magnet Permanen

Motor DC menjadi salah satu bagian yang penting dalam mesin CNC *milling*. Pada mesin CNC yang digunakan dalam penelitian ini, motor DC digunakan untuk menggerakkan *spindle*, *vice*, dan sumbu koordinat X, Y, dan Z. Motor DC adalah mesin listrik yang mengubah energi listrk arus searah menjadi energi mekanik. Salah satu jenis motor DC yaitu motor DC magnet permanen, dimana medan magnet dihasilkan oleh magnet permanen dan menghasilkan fluks konstan.

2.1.3.1 Prinsip Kerja [3]

Motor DC magnet permanen terdiri dari magnet permanen, kumparan jangkar, dan sikat (*brush*). Motor DC memiliki dua prinsip dasar yang melatarbelakangi prinsip kerja motor DC. Medan magnet menghasilkan magnet permanen yang memiliki nilai konstan, sedangkan komutator dan sikat berfungsi menyalurkan alur listrik dari sumber di luar motor ke dalam kumparan jangkar. Bagian-bagian motor DC ditunjukkan seperti pada Gambar 2.6.

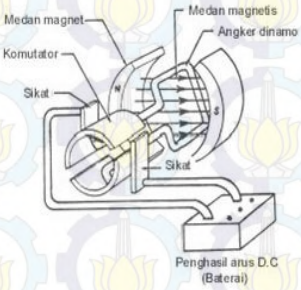


Gambar 2.6 Bagian-bagian Motor DC

Medan pada kumparan stator menghasilkan fluks (Φ) dari kutub utara (*U*) ke kutub selatan (*S*). Sikat arang menyentuh terminal kumparan rotor. Bila sikat arang dihubungkan pada satu sumber arus searah dengan tegangan *V*, maka arus *I* masuk ke terminal kumparan rotor di bawah kutub *U* dan keluar dari terminal di bawah kutub *S*. Dengan adanya fluks stator dan arus rotor, motor akan menghasilkan satu gaya *F* yang bekerja pada kumparan. Gaya ini dikenal dengan gaya Lorentz. Arah *F* yang ditunjukkan pada Gambar 2.6 menghasilkan torsi yang memutar rotor ke arah yang berlawanan dengan jarum jam. Kumparan yang membawa arus bergerak menjauhi sikat arang dan dilepas ke sumber luar. Kumparan berikutnya bergerak di bawah sikat arang dan membawa arus *I*. Dengan demikian, gaya *F* terus menerus diproduksi sehingga rotor berputar secara kontinyu.

2.1.3.2 Komponen Utama Motor DC [3]

Sebuah motor DC terdiri dari tiga penyusun utama seperti yang ditunjukkan pada Gambar 2.7.



Gambar 2.7 Komponen Utama Motor DC [4]

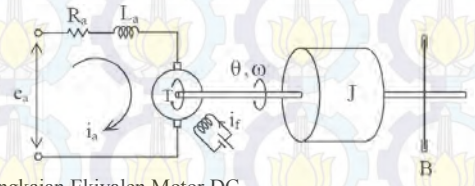
Ketiga komponen tersebut antara lain:

1. Kutub Medan
   Secara sederhana, dapat digambarkan bahwa interaksi dua kutub magnet akan menyebabkan perputaran pada motor DC. Motor DC memiliki kutub medan yang stasioner dan dinamo yang menggerakkan *bearing* pada ruang di antara kutub medan. Motor DC sederhana memiliki 2 kutub medan yaitu kutub utara dan kutub selatan.

2. Rotor
   Bila arus masuk menuju kumparan jangkar, maka arus ini menjadi elektromagnet. Rotor yang berbentuk silinder, dihubungkan ke as penggerak untuk menggerakkan beban. Untuk motor DC yang kecil, rotor berputar dalam medan magnet yang dibentuk oleh kutub-kutub, sampai kutub utara dan kutub selatan magnet berganti lokasi. Jika hal ini terjadi, arus berbalik merubah arah gerakan rotor.
3. Komutator
   Komponen ini terdapat pada motor DC dan berfungsi untuk menghasilkan tegangan searah dalam kumparan jangkar.

2.1.3.3 Model Matematika Motor DC [4]

Pada motor DC, terdapat 2 bagian yaitu bagian elektrik dan bagian mekanik. Rangkaian ekivalen motor DC ditunjukkan pada Gambar 2.8. Motor DC magnet permanen memiliki kumparan medan magnet yang bernilai konstan yang tidak dapat diatur arus medannya. Motor DC magnet permanen juga memiliki kumparan jangkar. Model matematika untuk *plant* motor DC dapat dijabarkan berdasarkan bagian elektrik dan mekanik.



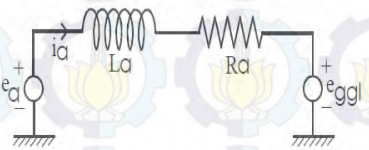
Gambar 2.8 Rangkaian Ekivalen Motor DC

dimana:

L_a = induktansi kumparan jangkar
R_a = resistansi kumparan jangkar
i_a = arus kumparan jangkar
if = arus medan
θ = perpindahan sudut poros motor
ω = perpindahan sudut poros motor
e_a = tegangan kumparan jangkar
e_{ggl} = tegangan gaya gerak listrik
J = momen inersia ekivalen motor dan beban pada arus motor

B = koefisien gesekan ekivalen motor dan beban pada poros motor

1. Rangkaian Kumparan Jangkar
   Rangkaian jangkar terdiri dari atas tahanan dan induktasi jangkar seperti yang ditunjukkan pada Gambar 2.9.



Gambar 2.9 Rangkaian Kumparan Jangkar

Persamaan differensial pada rangkaian jangkar adalah:

$$e_a(t) - e_{ggl}(t) = L_a \frac{di_a(t)}{dt} + R_a . i_a(t) \quad (2.1)$$

Transformasi Laplace dari Persamaan 2.1 adalah:

$$E_a(s) - E_{ggl}(s) = (L_a s + R_a) I_a(s) \quad (2.2)$$

$$I_a(s) = \frac{1}{(L_a s + R_a)} \left(E_a(s) - E_{ggl}(s) \right) \quad (2.3)$$

2. Torsi Motor
   Torsi T yang dihasilkan motor adalah berbanding lurus dengan hasil kali arus kumparan jangkar dan medan magnetik yang dihasilkan oleh penguat medan, yang berbanding lurus dengan arus medan atau dapat dirumuskan dengan:

$$B = \frac{\mu_f . n_f . i_f(t)}{2\pi l f} = K_B . i_f(t) \quad (2.4)$$

dimana K_B adalah konstanta medan magnet sehingga torsi T dapat ditulis sebagai berikut:

$$T = K_B i_f(t) l_a r_a n_a = K_{TM} i_f(t) i_a(t) \quad (2.5)$$

K_{TM} adalah konstanta torsi motor, karena arus medan i_f maka:

$$T = K_{TM} i_a(t) \quad (2.6)$$

Transformasi Laplace dari Persaman 2.5 adalah:

$$T(s) = K_{TM} I_a(t) \quad (2.7)$$

3. Momen Inersia

Pada bagian mekanik motor, terdapat persamaan kesetimbangan torsi motor dengan beban. Momen inersia dan torsi gesekan terletak pada bagian beban. Torsi yang dihasilkan motor bekerja terhadap inersia dengan gesekan viskos, sehingga:

$$T = J\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta}{dt} \quad (2.8)$$

atau

$$T = J\frac{d\omega(t)}{dt} + B\omega(t) \quad (2.9)$$

Transformasi Laplace dari Persaman 2.8 adalah:

$$T(s) = Js^2\Theta(s) + Bs\Theta(s) \quad (2.10)$$

$$\Theta(s) = \frac{1}{Js^2+Bs}T(s) \quad (2.11)$$

Persaman 2.10 digunakan untuk mengatur posisi motor.

Transformasi Laplace dari Persaman 2.8 adalah:

$$T(s) = Js\,\Omega(s) + B\Omega(s) \quad (2.12)$$

$$\Omega(s) = \frac{1}{Js+B}T(s) \quad (2.13)$$

Persaman 2.13 digunakan untuk mengatur posisi motor.

4. Tegangan Gaya Gerak Listrik

Besarnya tegangan *ggl* adalah perbandingan lurus dengan hasil kali arus medan dan keepatan sudut motor, yaitu:

$$e_{ggl}(t) = K.i_f(t).\frac{d\theta(t)}{dt} \quad (2.14)$$

dimana K adalah konstanta, karena arus medan konstan maka:

$$e_{ggl}(t) = K_g(t).\frac{d\theta(t)}{dt} \quad (2.15)$$

atau

$$e_{ggl}(t) = K_g.\omega(t) \quad (2.16)$$

dimana K_g adalah konstanta tegangan *ggl* balik.

Transformasi Laplace dari Persaman 2.15 adalah:

$$E_{ggl}(s) = K_g.s.\Theta(s) \quad (2.17)$$

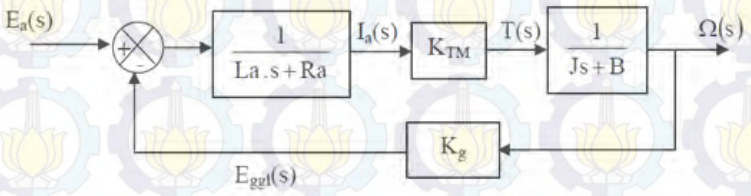
Persaman 2.15 digunakan untuk mengatur posisi motor.

Transformasi Laplace dari Persaman 2.16 adalah:

$$E_{ggl}(s) = K_g.\Omega(s) \tag{2.18}$$

Persaman 2.16 digunakan untuk mengatur posisi motor.

Untuk keperluan pengaturan kecepatan motor DC maka diagram blok total sistem diperoleh dengan menggabungkan Persamaan 2.3, 2.7, 2.13 dan 2.18. Diagram blok sistem pengaturan kecepatan motor DC ditunjukkan pada Gambar 2.10.



Gambar 2.10 Digram Blok Sistem Pengaturan Kecepatan Motor DC [4]

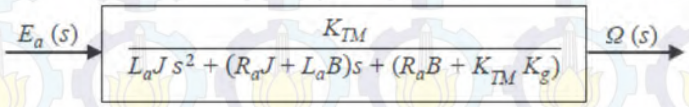
Fungsi alih lup tertutup $\Omega(s)/E_a(s)$ diperoleh sebagai berikut:

$$\frac{\Omega(s)}{E_a(s)} = \frac{\frac{K_{TM}}{(L_a s+R_a)(Js+B)}}{1+\frac{K_{TM}K_g}{(L_a s+R_a)(Js+B)}}$$

$$\frac{\Omega(s)}{E_a(s)} = \frac{K_{TM}}{(L_a s+R_a)(Js+B)+(K_{TM}K_g)}$$

$$\frac{\Omega(s)}{E_a(s)} = \frac{K_{TM}}{L_a\,Js^2+(R_a\,J+L_aB)s+(R_aB+K_{TM}K_g)} \tag{2.19}$$

Sehingga diagram blok pada Gambar 2.9dapat disederhanakan menjadi Gambar 2.10.



Gambar 2.11 Penyederhanaan Diagram Blok Sistem Pengaturan Kecepatan Motor DC [4]

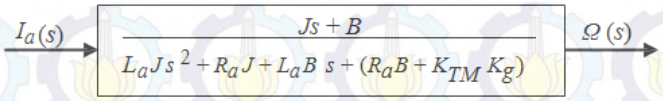
Selain mendapatkan fungsi alih untuk respon sistem, diagram blok motor DC yang ditunjukkan pada Gambar 2.10, juga dapat digunakan untuk mendapat fungsi alih untuk arus kumparan jangkar. Fungsi alih lup tertutup $I_a(s)/E_a(s)$diperoleh sebagai berikut:

$$\frac{\Omega(s)}{I_a(s)} = \frac{\frac{1}{(L_a s+R_a)}}{1+\frac{K_{TM}\,K_g}{(L_a s+R_a)(Js+B)}}$$

$$\frac{\Omega(s)}{I_a(s)} = \frac{Js+B}{(L_a s+R_a)(Js+B)+(K_{TM}K_g)}$$

$$\frac{\Omega(s)}{I_a(s)} = \frac{Js+B}{L_a\,Js^2+(R_a\,J+L_aB)s+(R_aB+K_{TM}K_g)} \qquad (2.20)$$

Sehingga diagram blok arus jangkar adalah:



Gambar 2.12 Digram Blok Respon Arus Jangkar

Dari Persamaan 2.19 dan 2.20 dapat disimpulkan bahwa fungsi alih untuk respon kecepatan dan arus motor DC memiliki nilai pembilang yang berbeda namun penyebutnya sama.

2.2 Identifikasi Sistem

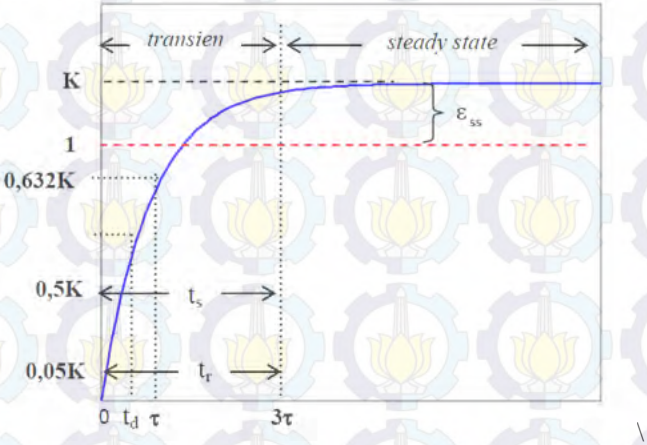
Model matematika diperlukan untukmmenggambarkan hubungan antara masukan dan keluaran sistem. Salah satu cara untuk mendapatkan model matematika sistem adalah dengan metode identifikasi. Metode identifikasi adalah suatu metode yang menggunakan hubungan data masukan dan keluaran yang selanjutnya dilakukan pengujian dan analisis dengan model pendekatan (asumsi model), sehingga dapat ditentukan nilai parameter secara analitis. Untuk mendapatkan respon sistem tersebut, dapat digunakan beberapa sinyal masukan, seperti sinyal *step, ramp, impulse*, dan sinusoidal.

Terdapat 2 macam identifikasi sistem yaitu identifikasi dinamis dan identifikasi statis. Pada identifikasi statis, sinyal yang diberikan berupa sinyal *step* yang konstan sampai sistem mencapai keadaan *steady state*. Pada identifikasi dinamis, sinyal yang digunakan

sebagai masukan merupakan sinyal acak, sehingga menghasilkan nilai keluaran yang acak pula.

2.2.1 Identifikasi Statis [4],[5]

Suatu sistem disebut orde satu atau pendekatan orde satu apabila dilihat secara grafik memiliki bentuk respon seperti pada Gambar 2.12.



Gambar 2.13 Karakteristik Respon Orde Satu [4]

Karaketristik respon orde satu dilihat berdasarkan respon system ketika diberi masukan berupa sinyal step. Karakteristik sistem orde satu dibedakan menjadi karakteristik respon transien dan *staeady state*. Model Matematika orde satu dirumuskan dengan:

$$G(s) = \frac{K}{\tau s+1} \tag{2.21}$$

Berdasarkan Persamaan 2.21, terdapat 2 parameter yaitu K yang menyatakan gain *overall* dan τ yang menyatakan *time constant*. Nilai K merupakan hasil perhitungan dari Persamaan 2.22:

$$K = \frac{Y_{ss}}{X_{ss}} \tag{2.22}$$

dimana Y_{ss} adalah keluaran sistem dan X_{ss} adalah masukan sistem saat kondisi *steady state*.

Terdapat 2 macam karakteristik respon pada sistem orde yaitu karakteristik respon transien dan *steady state*. Spesifikasi dari karakteristik respon transien pada sistem orde satu terdiri dari:

1. Spesifikasi Teoristis

   Time contant (τ) adalah waktu yang dibutuhkan respon mulai dari t = 0 sampai respon mencapai 63,2% dari respon *steady state*. *Time constant* menyatakan kecepatan respon sistem. Semakin kecil nilai τ maka semakin cepat respon sistem.

2. Spesifikasi Praktis

   a) *Settling time*/waktu tunak (t_s) adalah ukuran waktu yang menyatakan bahwa respon sistem telah masuk pada daerah *steady state*. Jika dihubungkan dengan nilai τ, maka dapat dirumuskan menjadi:

   $$t_s = (\pm 5\%) \approx 3\tau \quad (2.23)$$

   $$t_s = (\pm 2\%) \approx 4\tau \quad (2.24)$$

   $$t_s = (\pm 0{,}5\%) \approx 5\tau \quad (2.25)$$

   b) *Rise time*/waktu naik (t_r) adalah ukuran waktu yang menyatakan bahwa respon sistem telah naik 5% sampai 95% atau 10% sampai 90% dari nilai respon *steady state*.

   $$t_r = (5\% - 95\%) \approx \tau \ln 19 \quad (2.26)$$

   $$t_r = (10\% - 90\%) \approx \tau \ln 9 \quad (2.27)$$

   c) *Delay time*/waktu tunda (t_d) adalah waktu yang dibutuhkan respon mulai dari t = 0 sampai respon mencapai 50% dari nilai respon *steady state*. *Delay time* menyatakan besarnya faktor keterlambatan respon yang merupakan pengaruh dari proses *sampling*.

   $$t_d = \tau \ln 2 \quad (2.28)$$

Karakteristik respon *steady state* sistem orde satu dapat diukur berdasarkan kesalahan pada keadaan tunak atau disebut *error steady state* (e_{ss}), berikut adalah perhitungannya:

$$e_{ss} = Y_{ss} - X_{ss} \quad (2.29)$$

2.2.2 Metode Identifikasi Sistem Tanpa Osilasi [6]

Identifikasi sstem tanpa osilasi dengan masukan sinyal step dapat dilakuakan dengan beberapa metode. Pada subbab ini dijelaskan 7 metode, antara lain:

1. Metode Vítečková Orde 1
2. Metode Vítečková Orde 2
3. Metode Latzel
4. Metode Harriott
5. Metode Smith
6. Metode Strejc
7. Metode Sundaresan & Krishnaswamy

Untuk semua metode, terdapat *gain overall* yaitu *K*. Dimana *K* adalah hasil perhitungan keluaran *steady state* dibagi dengan masukan *steady state* yang dirumuskan sebegai berikut:

$$K = \frac{Y_{ss}}{X_{ss}} \tag{2.30}$$

dimana Y_{ss} adalah keluaran sistem dan X_{ss} adalah masukan sistem saat kondisi *steady state.*

2.2.2.1 Metode Vítečková Orde 1

Metode ini didekati dengan pendekatan sistem orde 1 dengan kemungkinan teradapat *delay time* . Fungsi alih untuk metode Vítečková Orde 1 seperti pada Persamaan 2.31.

$$G_{V1}(s) = \frac{K}{\tau_{V1}\, s+1} e^{-T_{dV1}\, s} \tag{2.31}$$

dimana T_{dV1} adalah *delay time* yang dirumuskan sebagai berikut:

$$T_{dV1} = 1{,}498\ t_{33} - 0{,}498\ t_{70} \tag{2.32}$$

dan τ_{V1} adalah *time constant* yang dirumuskan sebagai berikut:

$$\tau_{V1} = 1{,}245\ (t_{70} - t_{33}) \tag{2.33}$$

dengan t_{33} dan t_{70} adalah waktu saat respon sistem berada pada konsidisi33% dan 70% dari keluaran *steady state (*Y_{ss}*)*. Jika T_{dV1}bernilai negatif maka sistem tidak memiliki *delay time*.

2.2.2.2 Metode Vítečková Orde 1

Metode ini didekati dengan pendekatan sistem orde 2 dengan kemungkinan teradapat *delay time* . Fungsi alih untuk metode Vítečková Orde 2 seperti pada Persamaan 2.34.

$$G_{V2}(s) = \frac{K}{\tau_{V1}\, s+1} e^{-T_{dV2}\, s} \quad (2.34)$$

dimana T_{dV2} adalah *delay time* yang dirumuskan sebagai berikut:

$$T_{dV2} = 1{,}937\ t_{33} - 0{,}937\ t_{70} \quad (2.35)$$

dan τ_{V1} adalah *time constant* yang dirumuskan sebagai berikut:

$$\tau_{V2} = 0{,}794\ (t_{70} - t_{33}) \quad (2.36)$$

dengan t_{33} dan t_{70} adalah waktu saat respon system berada pada kondisi 33% dan 70% dari keluaran *steady state* (Y_{ss}). Jika T_{dV2}bernilai negatif maka system tidak memiliki *delay time*.

2.2.2.3 Metode Latzel

Fungsi alih untuk metode Latzel seperti pada Persamaan 2.37.

$$G_L(s) = \frac{K}{(\tau_L\, s+1)^n} \quad (2.37)$$

dimana τ_L adalah *time constant* yang dirumuskan sebagai berikut:

$$\tau_L = \alpha_{10}\, t_{10} + \alpha_{50}\, t_{50} + \alpha_{90}\, t_{90} \quad (2.38)$$

dengan t_{10}, t_{50} dan t_{90} adalah waktu saat respon system berada pada kondisi 10%, 50% dan 70% dari keluaran *steady state* (Y_{ss}). Jika T_{dV2}bernilai negatif maka system tidak memiliki *delay time*.

Untuk mendapatkan nilai α_{10}, α_{50} dan α_{90} terlebih dahulu mencari niali parameter μ yang dirumuskan dengan:

$$\mu = \frac{t_{10}}{t_{90}} \quad (2.39)$$

Setelah diperoleh nilai parameter μ, maka dapat diperoleh nilai parameter μ_a. Nilai parameter μ_a ditunjukkan pada Tabel 2.1 yang berupa tabel. Dari nilai μ_a yang ada pada tabel dapat diperoleh orde *n*, dan parameter α_{10}, α_{50} dan α_{90}.

Tabel 2.1 Parameter Metode Latzel [6]

μ_a	n	α_{10}	α_{50}	α_{90}	μ_a	n	α_{10}	α_{50}	α_{90}
0,137	2	1,880	0,596	0,257	0,456	11	0,142	0,094	0,065
0,174	2,5	1,245	0,460	0,216	0,472	12	0,128	0,086	0,060
0,207	3	0,907	0,374	0,188	0,486	13	0,116	0,079	0,056
0,261	4	0,573	0,272	1,150	0,499	14	0,106	0,073	0,053
0,304	5	0,411	0,214	0,125	0,512	15	0,097	0,068	0,050
0,340	6	0,317	0,176	0,108	0,523	16	0,090	0,064	0,047
0,370	7	0,257	0,150	0,095	0,533	17	0,084	0,060	0,045
0,396	8	0,215	0,130	0,085	0,543	18	0,078	0,057	0,042
0,418	9	0,184	0,115	0,077	0,552	19	0,073	0,054	0,040
0,438	10	0,161	0,103	0,070	0,561	20	0,069	0,051	0,039

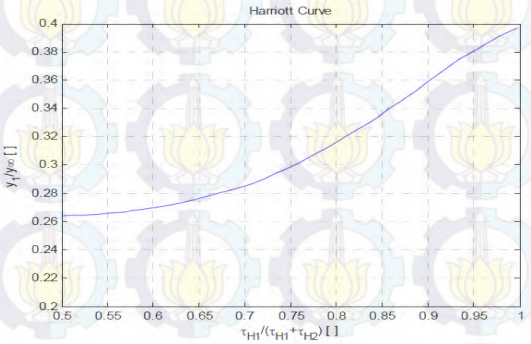
Setelah diperoleh nilai parameter n, α_{10}, α_{50} dan α_{90}, parameter τ_L dapat dihitung dan subtitusikan hasilnya pada Persamaan 2.37.

2.2.2.4 Metode Harriot

Metode ini didekati dengan pendekatan sistem orde 2 dengan kemungkinan teradapat *delay time* . Fungsi alih untuk metode Harriot seperti pada Persamaan 2.29.

$$G_H(s) = \frac{K}{(\tau_{H1}\, s+1)(\tau_{H2}\, s+1)} e^{T_{dH} s} \qquad (2.40)$$

dimana τ_{H1} dan τ_{H2} adalah *parameter yang dihitung dari pembacaan kurve Harriot* yang ditunjukkan pada Gambar 2.14.



Gambar 2.14 Kurva Harriot

Untuk mendapat parameter τ_{H1}, τ_{H2} dan T_{dH} dapat dilakukan prosedur sebagai berikut:

1. Mencari Jumlah τ_{H1} dan τ_{H2}
   Jumlah dari parameter τ_{H1} dan τ_{H2} dirumuskan sebagai berikut:

$$\tau_{H1} + \tau_{H2} = \frac{t_{73}}{1,3} \quad (2.41)$$

dengan t_{73} adalah waktu saat respon berada pada kondisi 73% dari keluaran *steady state* (*Yss*).

2. Mencari Parameter t_1 dan Y_1

   Parameter t_1 dibutuhkan untuk mendapat parameter Y_1, dimana t_1 dan Y_1 diperoleh dari respon sistem sebagai berikut:

   $$\tau_1 = \frac{\tau_{H1}+\tau_{H2}}{2} \quad (2.42)$$

3. Mencari Parameter pada Kurva Harriot

   Parameter yang diberikan pada kurva Harriot adalah parameter yang ada pada sumbu X dan Y. Parameter sumbu Y pada kurva Harriot diberikan dengan:

   $$\frac{y_1}{y_\infty}[\,] \text{ atau } \frac{Y_1}{Y_{ss}} \quad (2.43)$$

4. Mencari Parameter τ_{H1}

   Diasumsikan nilai parameter $\tau_{H1}\ /(\tau_{H1} + \tau_{H2})[\ ]$ pada kurva Harriot adalah *x*, kemudian subtitusikan Persamaan 2.

   $$\frac{\tau_{H1}}{\tau_{H1}+\tau_{H2}} = x \quad (2.44)$$

   $$\frac{\tau_{H1}}{t_{73}+1,3} = x \rightarrow \tau_{H1} = x\,.\frac{t_{73}}{1,3} \quad (2.45)$$

5. Mencari Parameter τ_{H2}

   Untuk mendapatkan parameter τ_{H2}, subtitusikan Persamaan 2.45 pada Persamaan 2.41.

   $$\tau_{H1} + \tau_{H2} = \frac{t_{73}}{1,3} \quad (2.46)$$

   $$\left(x.\frac{t_{73}}{1,3}\right) + \tau_{H2} = \frac{t_{73}}{1,3} \quad (2.47)$$

   $$\tau_{H2} = \frac{t_{73}}{1,3} - \left(x.\frac{t_{73}}{1,3}\right) \quad (2.48)$$

6. Mencari Parameter T_{dH}

   T_{dH}adalah *delay time* yang dirumuskan sebagai berikut:

   $$T_{dH} = 1,937t_{33} - 0,937t_{70} \quad (2.49)$$

   dengan t_{33} dan t_{70} adalah waktu saat respon berada pada kondisi 33% dan 70 % dari keluaran *steady state* (*Yss*). Jika T_{dH} bernilai kurang dari 0 atau negatif, maka sistem dianggap tidak memiliki *delay time.*

   Setelah diperoleh nilai parameter τ_{H1}, τ_{H2} dan T_{dH}, subtitusikan

hasilnya pada Persamaan 2.40.

2.2.2.5 Metode Smith

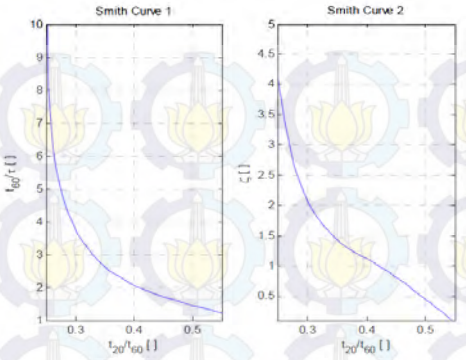
Metode ini didekati dengan pendekatan sistem orde 2 tanpa *delay time*. Fungsi alih untuk metode Smith seperti pada Persamaan 2.48.

$$G_L(s) = \frac{K}{(\tau_{SM1}\, s+1)(\tau_{SM2}\, s+1)} \tag{2.50}$$

dimana τ_{SM1} dan τ_{SM2} adalah parameter yang dihitung dari pembacaan kurva Smith 1 dan kurva Smith 2. Pada Gambar 2.14 menunjukkan kedua kurva Smith.

Untuk mendapat parameter τ_{SM1} dan τ_{SM2} dapat dilakukan prosedur sebagai berikut:

1. Mendapatkan Parameter τ Pada Kurva Smith 1
   Parameter sumbu X pada kurva Smith 1 dirumuskan dengan
   $$\frac{t_{20}}{t_{60}}[\,] \tag{2.51}$$
   dengan t_{20} dan t_{60} adalah waktu saat respon berada pada kondisi 20% dan 60% dari keluaran *steady state* (*Yss*). Setelah diperoleh nilai parameter t_{20}/t_{60} dapat ditarik garis sehingga diperoleh parameter $t_{60}\ /\tau$. Diasumsikan nilai parameter tersebut pada kurva Smith 1 adalah x, sehingga diperoleh nilai τ sebagai berikut:
   $$\frac{t_{60}}{\tau} = x \rightarrow \tau = \frac{t_{60}}{x} \tag{2.52}$$
2. Mencari Parameter ζ Pada Kurva Smith 2
   Parameter sumbu X pada kurva Smith 2 dirumuskan seperti Persamaan 2.49, kemudian dapat ditari garis sehingga diperoleh parameter ζ.
3. Mendapatkan Parameter τ_{SM1} dan τ_{SM2}
   Setelah nilai parameter τ dan ζ diperoleh, dapat diperoleh parameter τ_{SM1} dan τ_{SM2} yang dirumuskan sebagai berikut:
   $$\tau_{SM1} = \tau\,\zeta + \tau\sqrt{(\zeta^{\,2} - 1)} \tag{2.53}$$
   $$\tau_{SM2} = \tau\,\zeta - \tau\sqrt{(\zeta^{\,2} - 1)} \tag{2.54}$$



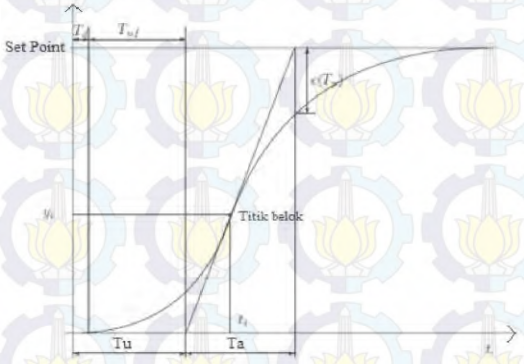
Gambar 2.15 Kurva Smith

Setelah diperoleh nilai parameter τ_{SM1} dan τ_{SM2}, subtitusikan hasilnya pada Persamaan 2.50.

2.2.2.6 Metode Strecj

Metode ini memerlukan penarikan garis singgung seperti yang ditunjukkan pada Gambar 2.15. Fungsi alih metode Strejc seperti pada Persamaan 2.55.

$$G_{ST}(s) = \frac{K}{(\tau_{ST}\, s+1)^n} e^{-T_{dST}s} \tag{2.55}$$



Gambar 2.16 Penarikan Garis Metode Strecj

dimana T_{dST} adalah *delay time*. Berikut prosedur metode Strecj untuk mendapatkan nilai parameter τ_{ST} dan T_{dST}:

Untuk mendapat parameter τ_{SM1} dan τ_{SM2} dapat dilakukan prosedur sebagai berikut:

1. Tentukan titik belok pada grafik respon yang diidentifikas.i
2. Tentukan garis singgung pada titik belok.
3. Dapatkan nilai T_u dan T_a.
4. Dapatkan nilai orde n, dimana nilai n sesuai dengan Tabel 2.2 ketika nilai T_u/T_a respon lebih besar daripada T_u/T_a pada tabel.
5. Dapatkan nilai τ_{ST} dengan menggunkan nilai T_a/τ_{ST} pada tabel.
6. Dapatkan nilai T_u', yang dirumuskan dengan:

$$T_u' = \left(\frac{T_u}{T_a}\right)_{tabel} \times T_a \tag{2.56}$$

7. Dapatkan nilai T_{dST} yang dirumuskan dengan:

$$T_{dS} = T_u - T_u' \tag{2.57}$$

8. Jika T_{dST} bernilai negatif, maka n salah, kembali ke langkah 5.
9. Subtitusikan parameter n, τ_{ST} dan T_{dST} pada Persamaan 2.55.

Tabel 2.2 Parameter Metode Strecj

n	T_a/τ_{ST}	T_u/τ_{ST}	T_u/T_a
1	1,000	0,000	0,000
2	2,718	0,282	0,104
3	3,695	0,805	0,218
4	4,463	1,425	0,319
5	5,119	2,100	0,410

2.2.2.7 Sundaresan & Krishnaswamy

Metode ini didekati dengan pendekatan sistem orde 1 dengan kemungkinan terdapat *delay time.* Fungsi alih untuk metode Sundaresan & Krishnaswamy seperti pada Persamaan 2.56.

$$G_{SK}(s) = \frac{K}{\tau_{SK}\, s+1} e^{-T_{dSK}S} \tag{2.58}$$

dimana T_{dSK} adalah *delay time* yang dirumuskan sebagai berikut:

$$T_{dSK}(s) = 1{,}3\, t_{35,3} - 0{,}29\, t_{85,3} \tag{2.59}$$

dimana τ_{SK} adalah *time constant* yang dirumuskan sebagai berikut:

$$\tau_{SK}(s) = 0{,}67\ (t_{85,3} - t_{35,3}) \quad (2.60)$$

dengan $t_{35,3}$ dan $t_{85,3}$ adalah waktu saat respon berada pada kondisi 35,3% dan 85,3 % dari keluaran *steady state* (*Yss*). Jika T_{dSK} bernilai kurang dari 0 atau negatif, maka sistem dianggap tidak memiliki *delay time.*

2.3 Kotroler PID

Kontroler PID adalah kontroler yang telah umum digunakan di industry. Kontroler in memiliki 3 parameter yaitu proporsional, integral dan derivatitif. Masing-masing parameter ini memiliki keunggulan dalam melakukan aksi kontrol. Keungulan proporsional yaitu untuk mempercepat rise time, keunggulan integral yaitu memperkecil error steady state dan keunggulan derivatif yaitu memperkecil osilasi dan *overshoot*.

Parameter P bergantung pada nilai kesalahan saat ini. Parameter I bergantung pada nilai kesalahan waktu sebelumnya. Parameter D bergantung pada prediksi kesalahan masa depan.

Ketiga kontroler ini digabungkan untuk menghasilkan sinyal kontrol *u(t)*sepertipada Persamaan 2.90.

$$u(t) = K_p \left[e(t) + \frac{1}{\tau_i} \int e(t)dt + \tau_d \frac{de(t)}{dt} \right] \quad (2.61)$$

Dalam bentuk fungsi alih menjadi seperti berikut:

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) \quad (2.62)$$

Kontroler PID memiliki parameter control yang harus disesuaikan dengan respon yang diinginkan pada suatu model *plant.*Pada system nonlinear, kontroler ini kurang handal sehingga diperlukan mekanisme penalaan parameter kontroler PID yang dapat menyesuaikan dengan keadaan sistem.

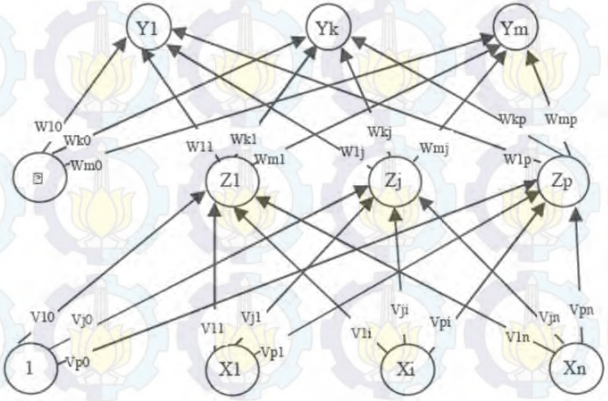
2.4 Jaringan Syaraf Tiruan [7]

Jaringan syaraf tiruan (JST) atau pada umumnya disebut *neural network* (NN) adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan syaraf manusia. JST merupakan sistem adaptif yang dapat merubah strukturnya untuk memecahkan masalah berdasarkan informasi ekternal maupun internal yang mengalir melalui jaringan tersebut.

Secara sederhana, JST adalah sebuah pemodelan data statistik non-linier. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan output untuk menemukan pola-pola pada data.

Salah satu model Jaringan Syaraf Tiruan adalah *backpropagation*. Seperti halnya model JST yang lain, *backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa walaupun tidak sama dengan pola yang dipakai selama pelatihan.

Arsitektur *backpropagation* dengan *n* buah masukan (ditambah satu bias), dan sebuah layar tersembunyi yang terdiri dari *p* unit (ditambah satu bias), dan juga *m* buah unit keluaran, terdapat pada gambar 2.17.



Gambar 2.17 Arsitektur *NN Backpropagation*

Gambar 2.15 dapat terlihat bahwa V_{ji} adalah bobot garis dari unit masukan X_i ke unit layar tersembunyi Z_j (V_{j0} adalah bobot garis yang menghubungkan bias di unit masukan ke unit layar tersembunyi Z_j). Sedangkan W_{kj} adalah bobot dari unit layer tersembunyi Z_j ke unit keluaran Y_k (W_{k0} adalah bobot dari bias dilayer tersembunyi ke unit keluaran Y_k).

Fungsi aktivasi dalam *backpropagation* yang dipakai harus memenuhi beberapa syarat yaitu: kontinu, terdiferensial dengan mudah

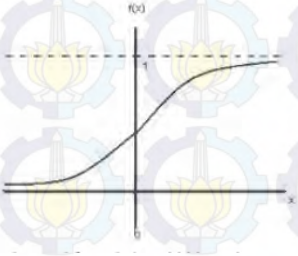
dan merupakan fungsi yang tidak turun. Salah satu fungsi aktivasi adalah fungsi sigmoid biner yang memiliki range (0,1).

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.63}$$

dengan turunan

$$f'(x) = f(x)\big(1 - f(x)\big) \tag{2.64}$$

Grafik fungsinya tampak pada gambar 2.16.



Gambar 2.18 Grafik Fungsi Aktivasi Sigmoid Biner

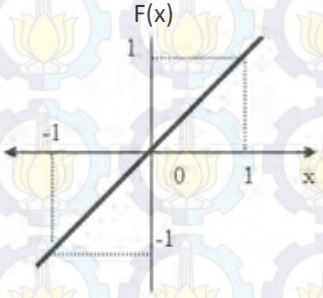
Fungsi aktivasi yang sering digunakan selain fungsi aktivasi sigmoid adalah fungsi aktivasi linier. Fungsi linier akan membawa nilai input sebanding dengan nilai output. fungsi ini digambarkan sebagai berikut:

$$f(x) = \lambda x \tag{2.65}$$

Dengan fungsi turunan

$$f'(x) = \lambda \tag{2.66}$$

Grafik fungsinya tampak pada gambar 2.17.



Gambar 2.19 Grafik Fungsi Aktivasi Linier

Pelatihan *backpropagation* meliputi 3 fase. Fase pertama adalah fase maju atau perhitungan maju. Fase kedua adalah propagasi mundur. Fase ketiga adalah perubahan nilai bobot.

Berikut adalah langkah – langkah melakukan perhitungan JST.

a) Langkah 0: inisialisasi semua bobot dengan bilangan acak kecil.
b) Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-9.
c) Langkah 2: Setiap pasang data pelatihan, lakukan langkah 3-8.
Fase 1: Perhitungan Maju.
d) Langkah 3: Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi diatasnya.
e) Langkah 4: Hitung semua keluaran di unit tersembunyi $Z_j(j = 1,2, \dots . p)$

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i \, v_{ji} \quad (2.67)$$

$$z_j = \text{f}(z\_net_j) = \frac{1}{1+e^{-z\_net_j}} \quad (2.68)$$

f) Langkah 5: Hitung semua keluaran jaringan di unit $y_k(k = 1,2, \dots . m)$

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j \, w_{kj} \quad (2.69)$$

$$y_k = \text{f}(k\_net_j) = \frac{1}{1+e^{-y\_net_k}} \quad (2.70)$$

Fase 2 : Perhitungan Mundur.

g) Langkah 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran $y_k(k = 1,2, \dots . m)$

$$\delta_k = (t_k - y_k) f'(y_{net_k}) = (t_k - y_k) y_k (1 - y_k) \quad (2.71)$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar pada langkah 7.

Menghitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju percepatan α.

$$\Delta w_{kj} = \alpha \, \delta_k \, z_j \; ; \; k = 1,2, \dots, m \; ; j = 0,1, \dots . p \quad (2.72)$$

h) Langkah 7: Hitung δ unit tersembunyi berdasarkan kesalahan disetiap unit tersembunyi $Z_j(j = 1,2, \dots . p)$

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k \, w_{kj} \quad (2.73)$$

Faktor δ unit tersembunyi :

$$\delta_j = \delta\_net_j \, f'(z_{net_j}) = \delta\_net_j z_j (1 - z_j) \quad (2.74)$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti untuk merubah bobot v_{ji}.

$$\Delta v_{ji} = \alpha \, \delta_j \, x_i \; ; \; j = 1,2, \dots, p \; ; i = 0,1, \dots . n \quad (2.75)$$

i) Langkah 8: Hitung semua perubahan bobot. Perubahan bobot baris yang menunju ke unit keluaran:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj}$$
$$(k = 1,2, \ldots . m\ ; j = 0,1, \ldots p) \qquad (2.75)$$

Perubahan bobot baris yang menuju ke unit tersembunyi:

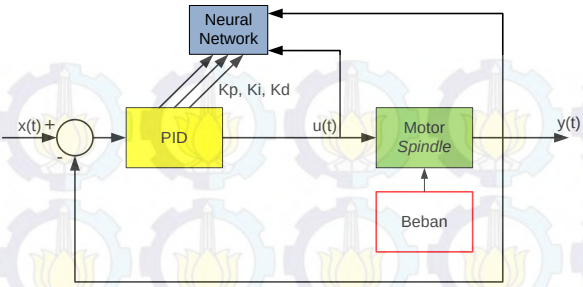
$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji}$$
$$(j = 1,2, \ldots , p\ ; i = 0,1, \ldots . n) \qquad (2.76)$$

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai untuk menentukan keluaran jaringan.

2.5 Kontroler *Neuro – PID*

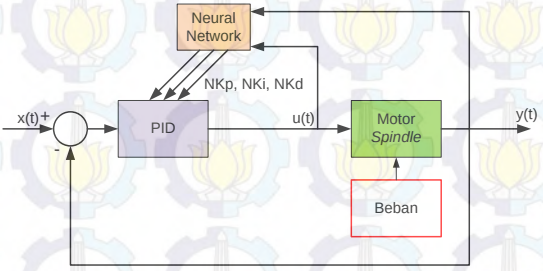
Kontroler PID merupakan kontrol lup tertutup yang sering digunakan dalam proses kontrol, namun kontroler ini memiliki kelemahan dalam penentuan parameter yang sesuai ketika terjadi perubahan dinamika proses, sehingga kurang efektif dan efisien dalam penerapan sistem kontrol. Oleh karena itu, diperlukan mekanisme penalaan parameter kontroler PID yang dapat menyesuaikan dengan perubahan parameter *plant* dan kondisi operasi sistem.

Untuk membuat kontroler *Neuro-PID* dilakukan dua tahap seperti ditunjukkan pada Gambar 2.20 dan 2.21, dimana tahap pertama adalah merancang kontroler PID dengan menentukan nilai K_p, K_i dan K_d fix lalu nilai tersebut dijadikan sebagai acuan dalam proses belajar *NN* yaitu sebagai tahap revisi bobot seperti pada persamaan 2.71. Setelah *NN* mampu belajar mencapai atau mendekati nilai K_p, K_i dan K_d yang diharapkan, maka bobot terbaik pun didapat untuk selanjutnya digunakan dalam proses *mapping*. Pada tahap *mapping*, input *NN* sama seperti saat proses belajar yaitu sinyal kontrol dan output sistem. Sedangkan output dari *NN* merupakan nilai NK_p, NK_i dan NK_d yang digunakan untuk memberikan sinyal kontrol pada *plant*.



Gambar 2.20 Diagram Blok Proses *Learning Neural Network*

Pada Gambar 2.20 terlihat bahwa nilai K_p, K_i dan K_d pada proses *learning Neural Network* diberikan dari hasil perancangan PID. Setelah itu pada blok *Neural Network* dibuat algoritma yang digunakan untuk proses proses belajar seperti yang telah dijelaskan dalam langkah-langkah perhitungan *Neural Network*.



Gambar 2.21 Diagram Blok Proses *Mapping Neural Network*

Pada Gambar 2.21 terlihat bahwa *Neural Network* menghasilkan nilai K_p, K_i dan K_d yang baru. Proses dilakukan setelah mendapatkan bobot terbaik dari proses learning. Jika bobot yang diperoleh benar-benar merupakan bobot yang tepat, maka nilai NK_p, NK_i dan NK_d yang dihasilkan dari *NN* akan persis atau mendekati nilai parameter PID yang telah diberikan saat proses *learning*. Hal ini akan membuat sinyal kontrol yang dihasilkan sesuai dengan kebutuhan sistem untuk mengatur *plant*.

BAB III
PERANCANGAN SISTEM

Bab 3 membahas tentang identifikasi sistem, fungsi alih sistem, validasi sistem, idetifikasi sistem untuk masing-masing kondisi beban, perancangan kontroler PID, perancangan kontroler *Neural Network*, perancangan proses *learning Neural Network* dan perancangan proses *mapping Neural Network*.

3.1 Identifikasi Sistem

Dalam tahapan ini langkah pertama yang akan dilakukan adalah identifikasi sistem dari motor *spindle*. Karena *plant* CNC sedang mengalami kerusakan, maka untuk menunjang pembuatan pemodelan matematika sistem diperoleh dari Tugas Akhir yang pernah dikerjakan oleh Nindita Suryaditya Putri dengan judul "Pengaturan Proses *Face Millin*g pada Mesin *Computer Numerical Control* (CNC) Dengan Kontroler *Fuzzy-PID*".

Identifikasi sistem diperlukan untuk mendapatkan model matematika dari motor *spindle*. Identifikasi yang dilakukan adalah identifikasi statis dengan melihat respon dari kecepatan referensi yang diberikan. Respon sistem yang diidentifikasi adalah pada saat kondisi beban minimal, nominal dan maksimal. Respon sistem akan berubah-ubah sesuai dengan kecepatan dan kondisi pembeban yang berubah-ubah pula. Pengambilan data dilakukan pada saat CNC melakukan proses *feeding* (pemakanan) untuk benda kerja berbahan plastik jenis polietilen yang Gambar 3.1. Proses *feeding* dengan gerakan arah horizontal diberikan kecepatan referensi sebesar 2000 rpm. Respon diperoleh saat kecepatan berubah dari 2000 rpm menjadi 3000 rpm. Pada keadaan *feedrate* yang berbeda akan menghasilkan respon sistem yang berbeda pula. Kondisi pembebanan diberikan seperti berikut:

Beban minimal : saat tidak menyentuh benda kerja
Beban nominal : saat *feedrate* 250 mm/min
Beban maksimal : saat *feedrate* 500 mm/min

Gambar 3.1 Benda Kerja Plastik Polietilen

Selanjutnya melakukan identifikasi berdasarkan paper dengan penulis Ing. Pavel Jakoubek, terdapat 7 metode identifikasi yang dapat digunakan unuk sistem tanpa osilasi dengan masukan respon step. Setelah menperoleh fungsi alih sistem, lalu dari ketujuh metode ini dibandingkan untuk mendapat model yang mendekati model *plant* dengan ISE (*Integral Square Error)* terkecil.

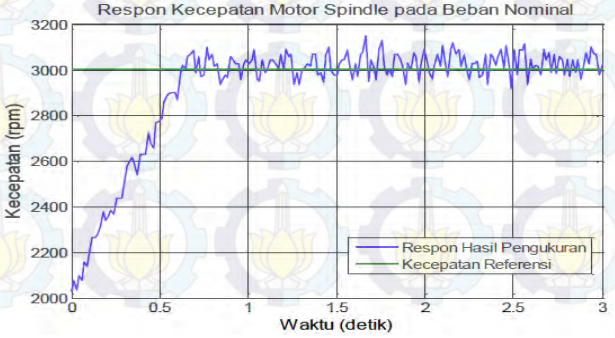
3.1.1 Fungsi Alih Sistem

Untuk mendapatkan fungsi alih sistem, respon yang digunakan adalah respon saat beban nominal. Respon kecepatan motor *spindle* pada kondisi beban nominal ditunjukkan pada Gambar 3.2. Pada respon tersebut, diperoleh nilai keluaran *steady state* pada kecepatan 3044 rpm. Nilai tersebut diperoleh dari rata-rata seluruh data keluaran sistem saat *steady state*. Berdasarkan respon pada Gambar 3.2, diperoleh gain overall respon pada Persamaan 3.1.

$$Y_{ss} = 3044 \tag{3.1}$$

$$X_{ss} = 3000 \tag{3.2}$$

$$K = \frac{Y_{ss}}{X_{ss}} = \frac{3044}{3000} = 1{,}014 \tag{3.3}$$



Gambar 3.2 Respon Kecepatan Motor *Spindle* Saat Beban Nominal

1. Metode Vítečková Orde 1
   Berdasarkan respon pada Gambar 3.2, diperoleh:

   $t_{33} = 0{,}212$ detik (3.4)

   $t_{70} = 0{,}486$ detik (3.5)

   sehingga diperoleh parameter:

   $T_{dV1} = (1{,}4983 \times 0{,}212) - (0{,}498 \times 0{,}486) = 0{,}0755$ (3.6)

   $T_{V1} = 0{,}2145 \times (0{,}486 - 0{,}212) = 0{,}3411$ (3.7)

   Fungsi alih untuk metode Vítečková Orde 1 adalah:

   $$G_{V1}(s) = \frac{1{,}014}{0{,}3411\,s+1} e^{-0.0755\,s} \quad (3.8)$$

2. Metode Vítečková Orde 2
   Parameter yang diperoleh dari respon pada Gambar 3.2 sebagai berikut:

   $T_{dV2} = (1{,}937 \times 0{,}212) - (0{,}937 \times 0{,}486) = -0{,}045$ (3.9)

   $T_{V2} = 0{,}794 \times (0{,}486 - 0{,}212) = 0{,}2176$ (3.10)

   Parameter T_{V2} bernilai negatif, sehingga parameter *time delay* T_{V2} tidak dianggap. Fungsi alih untuk metode Vítečková Orde 2 adalah:

   $$G_{V2}(s) = \frac{1{,}014}{(0{,}2176\,s+1)^2} \quad (3.11)$$

   $$G_{V2}(s) = \frac{1{,}014}{0{,}0473\,s^2+0{,}4351\,s+1} \quad (3.12)$$

3. Metode Latzel
   Berdasarkan respon pada Gambar 3.2, diperoleh:

   $t_{10} = 0{,}082$ detik (3.13)

   $t_{50} = 0{,}353$ detik (3.14)

   Sehingga diperoleh parameter:

   $$\mu = \frac{0{,}082}{0{,}593} = 0{,}1383 \quad (3.15)$$

   Setelah diperoleh parameter μ lalu dilakukan pendekatan sesuai dengan Tabel 2.1 untuk mendapatkan nilai μ_a. Sehingga diperoleh parameter sebagai berikut:

   $\mu_a = 0{,}137$ (3.16)

   $n = 2$ (3.17)

$$\alpha_{10} = 1{,}88 \quad (3.18)$$

$$\alpha_{50} = 0{,}596 \quad (3.19)$$

$$\alpha_{90} = 0{,}257 \quad (3.20)$$

$$\tau_L = (1{,}88 \times 0{,}082) + (0{,}596 \times 0.353) + (0{,}257 \times 0{,}593 = 0{,}5169 \quad (3.21)$$

Fungsi alih untuk metode Latzel adalah:

$$G_L(s) = \frac{1{,}014}{(0{,}5169\, s+1)^2} \quad (3.22)$$

$$G_L(s) = \frac{1{,}014}{0{,}2672\, s^2+1{,}0334\, S+1} \quad (3.23)$$

4. Metode Harriot

Berdasarkan respon pada Gambar 3.2, diperoleh:

$$t_{33} = 0{,}212 \text{ detik} \quad (3.24)$$

$$t_{70} = 0{,}486 \text{ detik} \quad (3.25)$$

$$t_{73} = 0{,}503 \text{ detik} \quad (3.26)$$

Sehingga diperoleh parameter:

$$T_{dH} = (1{,}937 \times 0{,}212) - (0{,}937 \times 0{,}486 = -0{,}045 \quad (3.27)$$

$$\tau_{H1} + \tau_{H2} = \frac{0{,}503}{1{,}3} = 0{,}3869 \quad (3.28)$$

$$t_1 = \frac{\tau_{H1} + \tau_{H2}}{2} = 0{,}135 \rightarrow y_1 = 2315 \quad (3.29)$$

Berdasarkan kurva Harriot pada Gambar 2.11, diperoleh nilai $\tau_{H1}/(\tau_{H1} + \tau_{H2})$:

$$\frac{y_1}{Y_{ss}} = \frac{2315}{3044} = 0{,}3017 \rightarrow \frac{\tau_{H1}}{(\tau_{H1}+\tau_{H2})} = 0{,}7568 \quad (3.30)$$

Sehingga nilai τ_{H1} dan τ_{H2} adalah:

$$\frac{\tau_{H1}}{(\tau_{H1}+\tau_{H2})} = 0{,}7568 \quad (3.31)$$

$$\frac{\tau_{H1}}{0{,}3869} = 0{,}7568 \quad (3.32)$$

$$\tau_{H1} = 0{,}2928 \quad (3.33)$$

$$\tau_{H1} + \tau_{H2} = 0{,}3869 \rightarrow \tau_{H2} = 0{,}3869 - 0{,}2928 = 0{,}0941 \quad (3.34)$$

Parameter *delay* T_{dH} bernilai negatif, sehingga nilai *delay* T_{dH} tidak dianggap. Fungsi alih untuk metode Harriot adalah:

$$G_H(s) = \frac{1{,}014}{(,0.2928\,s+1)(0{,}0941\,s+1)} \tag{3.35}$$

$$G_H(s) = \frac{1.014}{0{,}0276\,s^2 + 0{,}3869\,s+1} \tag{3.36}$$

5. Metode Smith

   Berdasarkan respon pada Gambar 3.2, diperoleh:

$$t_{20} = 0{,}194 \text{ detik} \tag{3.37}$$

$$t_{60} = 0{,}473 \text{ detik} \tag{3.38}$$

   Berdasarkan kurva Smith pada Gambar 2.12, diperoleh nilai τ_{60}/τ:

$$\frac{t_{20}}{t_{60}} = \frac{0{,}194}{0{,}473} = 0{,}4101 \rightarrow \frac{t_{60}}{\tau} = 1{,}9804 \tag{3.39}$$

   Berdasarkan kurva Smith pada Gambar 2.12, diperoleh nilai ξ:

$$\frac{t_{20}}{t_{60}} = \frac{0{,}194}{0{,}473} = 0{,}4101 \rightarrow \zeta = 1{,}0543 \tag{3.40}$$

$$\frac{t_{60}}{\tau} = 1{,}9804 \rightarrow \tau = 0{,}2388 \tag{3.41}$$

$$\tau_{SM1} = (0{,}2388 \times 1{,}0543) + 0{,}2388\sqrt{1{,}0543^2 - 1} = 0{,}3316 \tag{3.42}$$

$$\tau_{SM2} = (0{,}2388 \times 1{,}0543) - 0{,}2388\sqrt{1{,}0543^2 - 1} = 0{,}172 \tag{3.43}$$

   Fungsi alih untuk metode Hariot adalah:

$$G_{SM}(s) = \frac{1{,}014}{(0{,}3316\,s+1)(0{,}172\,s+1)} \tag{3.44}$$

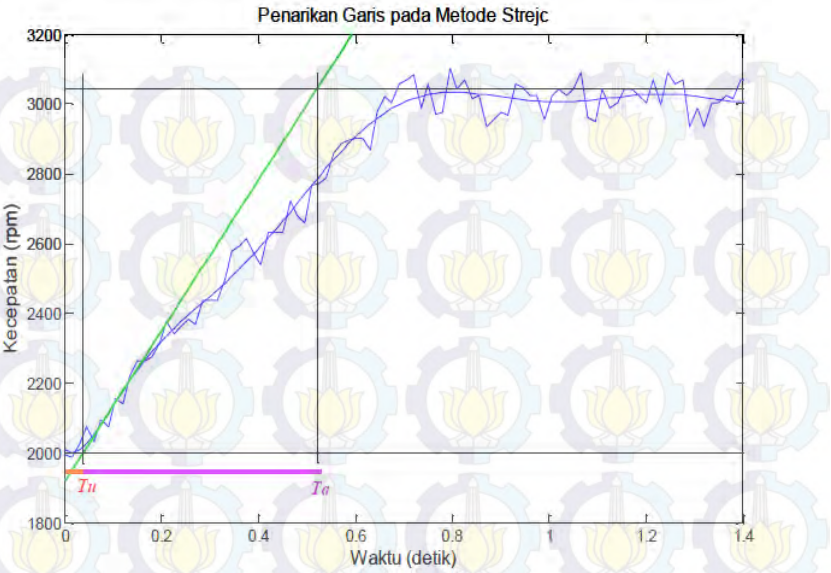
$$G_{SM}(s) = \frac{1{,}014}{0{,}057\,s^2 + 0{,}5036\,s+1} \tag{3.45}$$

6. Metode Strejc

   Metode Strejc memerlukan penarikkan garis seperti pada Gambar 3.3 untuk mendapatkan parameter fungsi alih. Berdasarkan Gambar 3.3 diperoleh:

$$T_u = 0{,}035 \text{ detik} \tag{3.46}$$

$$T_a = 0{,}525 \text{ detik} \tag{3.47}$$



Gambar 3.3 Penarikan Garis pada Metode Strecj

Sehingga diperoleh parameter:

$$\frac{T_u}{T_a} = \frac{0{,}035}{0{,}525} = 0{,}068 \tag{3.48}$$

Setelah diperoleh nilai parameter T_u/T_a, dilakukan pendekatan sesuai Tabel 2.2 untuk mendapat nilai T_u/T_a yang sesuai dengan prosedur metode Strejc.Sehingga diperoleh parameter sebagai berikut:

$$\left(\frac{T_u}{T_a}\right)_{tabel} = 0 \tag{3.49}$$

$$n = 1 \tag{3.50}$$

$$\frac{T_a}{\tau_{ST}} = 1 \rightarrow \tau_{ST} = \frac{T_a}{1} = \frac{0{,}525}{1} = 0{,}525 \tag{3.51}$$

$$T_u{}' = \left(\frac{T_u}{T_a}\right)_{tabel} \times T_a = 0 \times 0{,}525 = 0 \tag{3.52}$$

$$T_{dST} = T_u - T_u{}' = 0{,}035 - 0 = 0{,}035 \tag{3.53}$$

Fungsi alih untuk metode Strejc adalah:

$$G_{ST}(s) = \frac{1{,}014}{0{,}525\ s+1} e^{-0{,}035\ s} \quad (3.54)$$

7. Metode Sundaresan & Krishnaswamy
   Berdasarkan respon pada Gambar 3.2, diperoleh:

$$t_{35{,}3} = 0{,}23 \text{ detik} \quad (3.55)$$

$$t_{85{,}3} = 0{,}587 \text{ detik} \quad (3.56)$$

Sehingga diperoleh parameter:

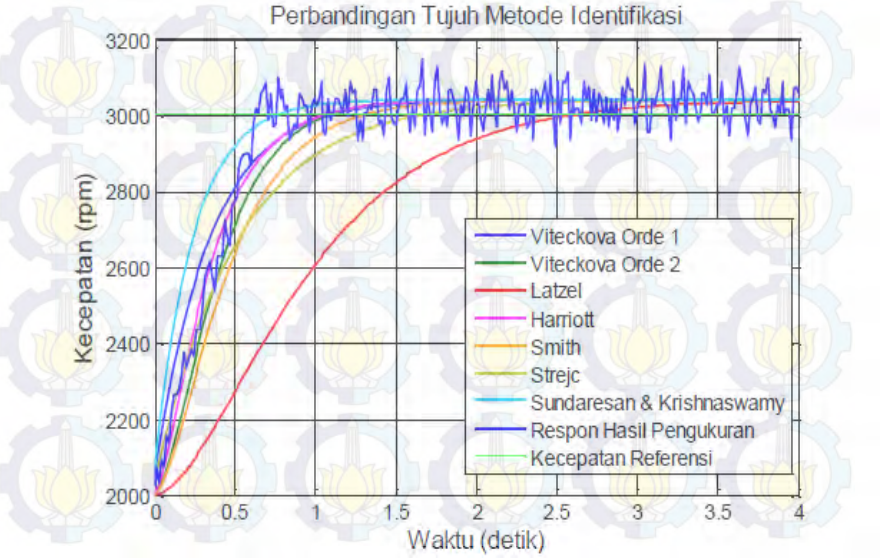
$$T_{dSK} = (1{,}3 \times 0{,}23) - (0{,}29 \times 0{,}587) = 0{,}1288 \quad (3.57)$$

$$T_{SK} = 0{,}67 \times (0{,}587 - 0{,}23) = 0{,}2392 \quad (3.58)$$

Fungsi alih untuk metode Sundaresan & Krishnaswamy adalah:

$$G_{SK}(s) = \frac{1{,}014}{(0{,}2392\ s+1)} e^{-0{,}1288\ s} \quad (3.59)$$

Setelah diperoleh fungsi alih sistem pada beban nominal masing-masing metode, dilakukan simulasi untuk mengetahui respon sistem dengan pendekatan terbaik terhadap hasil pengukuran. Grafik perbandingan respon sistem pada masing-masing metode ditunjukkan pada Gambar 3.4.



Gambar 3.4 Perbandingan Respon Sistem Setiap Metode

3.1.2 Validasi Sistem

Pemilihan metode terbaik dilakukan dengan validasi. Masing-masing fungsi alih yang telah diperoleh dari keenam metode tersebut diuji dan dibandingkan hasilnya dengan data hasil pengukuran. Metode validasi yang digunakan adalah ISE (*Integral Square Error*). Semakin kecil nilai ISE, maka semakin baik fungsi alih yang dibuat.

Hasil perhitungan ISE dari masing-masing metode diberikan pada Tabel 3.1.

Tabel 3.1 Fungsi Alih Masing-masing Metode

Metode	Fungsi Alih	ISE
Metode Vítečková Orde 1	$G_{V1}(s) = \frac{1{,}014}{0{,}3411\,s + 1} e^{-0.0755\,s}$	$2{,}3161 \times 10^6$
Metode Vítečková Orde 2	$G_{V2}(s) = \frac{1{,}014}{0{,}0473\,s^2 + 0{,}4351\,s + 1}$	$2{,}3100 \times 10^6$
Latzel	$G_L(s) = \frac{1{,}014}{0{,}2672\,s^2 + 1{,}0344\,s + 1}$	$1{,}9001 \times 10^7$
Harriot	$G_H(s) = \frac{1.014}{0{,}0276\,s^2 + 0{,}3869\,s + 1}$	$1{,}9959 \times 10^6$
Smith	$G_{SM}(s) = \frac{1{,}014}{0{,}057\,s^2 + 0{,}5036\,s + 1}$	$3{,}1595 \times 10^6$
Strejc	$G_{ST}(s) = \frac{1{,}014}{0{,}525\,s + 1} e^{-0{,}035\,s}$	$3{,}2588 \times 10^6$
Sundaresan & Krishnaswamy	$G_{SK}(s) = \frac{1{,}014}{(0{,}2392\,s + 1)} e^{-0{,}1288\,s}$	$3{,}4621 \times 10^6$

Berdasarkan fungsi alih dan perhitungan ISE yang ditunjukkan pada Tabel 3.1, dapat disimpulkan bahwa fungsi alih yang digunakan adalah fungsi alih yang didapat dari identifikasi dengan metode Harriot. Karena memiliki nilai ISE paling kecil yaitu 1,9959 x 10^6.

3.1.3 Identifikasi Beban Minimal

Identifikasi model sistem dilakukan saat kondisi beban minimal yaitu ketika motor *spindle* melakukan *feeding* namun tanpa menyentuh benda kerja atau tanpa beban. Identifikasi beban minimal dilakukan sebanyak lima kali percobaan dan menghasilkan fungsi alih serta ISE seperti yang ditunjukkan pada Tabel 3.2.

Tabel 3.2 Hasil Identifikasi Beban Minimal

No.	Fungsi Alih	ISE
1	$G_H(s) = \frac{1{,}0186}{0{,}0093\ s^2 + 0{,}2146\ s + 1}$	$1{,}7633 \times 10^6$
2	$G_H(s) = \frac{1{,}014}{0{,}0101\ s^2 + 0{,}2131\ s + 1} e^{-0{,}0076\ s}$	$1{,}6816 \times 10^6$
3	$G_H(s) = \frac{1{,}0136}{0{,}007\ s^2 + 0{,}2008\ s + 1}$	$1{,}7969 \times 10^6$
4	$G_H(s) = \frac{1.014}{0{,}0099\ s^2 + \ 0{,}2046\ s + 1}$	$1{,}7258 \times 10^6$
5	$G_H(s) = \frac{1{,}0132}{0{,}0104\ s^2 + 0{,}2062\ s + 1}$	$1{,}7786 \times 10^6$

Berdasarkan fungsi alih dan perhitungan ISE yang ditunjukkan pada Tabel 3.2, dapat disimpulkan bahwa fungsi alih yang digunakan adalah fungsi alih nomor 2 dengan ISE paling kecil yaitu 1,6816 x 10^6. Fungsi alih sistem pada beban minimal yaitu:

$$G_H(s) = \frac{1{,}014}{0{,}0101\ s^2 + 0{,}2131\ s + 1} e^{-0{,}0076\ s} \tag{3.60}$$

3.1.4 Identifikasi Beban Nominal

Identifikasi model sistem dilakukan saat kondisi nominal yaitu ketika motor *spindle* melakukan *feeding* terhadap benda kerja dengan *feedrate* 250 mm/min. Identifikasi beban nominal dilakukan sebanyak lima kali percobaan dan menghasilkan fungsi alih serta ISE seperti yang ditunjukkan pada Tabel 3.3.

Tabel 3.3 Hasil Identifikasi Beban Nominal

No.	Fungsi Alih	ISE
1	$G_H(s) = \frac{1{,}014}{0{,}0276\,s^2 + 0{,}3869\,s + 1}$	$2{,}0156 \times 10^6$
2	$G_H(s) = \frac{1{,}014}{0{,}0401\,s^2 + 0{,}4208\,s + 1}$	$1{,}7084 \times 10^6$
3	$G_H(s) = \frac{1{,}014}{0{,}0401\,s^2 + 0{,}4208\,s + 1}$	$1{,}8635 \times 10^6$
4	$G_H(s) = \frac{1.014}{0{,}0391\,s^2 + 0{,}4138\,s + 1}$	$1{,}8789 \times 10^6$
5	$G_H(s) = \frac{1{,}014}{0{,}0396\,s^2 + 0{,}4385\,s + 1}$	$1{,}8492 \times 10^6$

Berdasarkan fungsi alih dan perhitungan ISE yang ditunjukkan pada Tabel 3.3, dapat disimpulkan bahwa fungsi alih yang digunakan adalah fungsi alih nomor 2 dengan ISE paling kecil yaitu 1,7084 x 10^6. Fungsi alih sistem pada beban nominal yaitu:

$$G_H(s) = \frac{1{,}014}{0{,}0401\,s^2 + 0{,}4208\,s + 1} \tag{3.61}$$

3.1.5 Identifikasi Beban Maksimal

Identifikasi model sistem dilakukan saat kondisi maksimal yaitu ketika motor *spindle* melakukan *feeding* terhadap benda kerja dengan *feedrate* 500 mm/min. Identifikasi beban nominal dilakukan sebanyak lima kali percobaan dan menghasilkan fungsi alih serta ISE seperti yang ditunjukkan pada Tabel 3.4.

Tabel 3.4 Hasil Identifikasi Beban Maksimal

No.	Fungsi Alih	ISE
1	$G_H(s) = \frac{1{,}0209}{0{,}1495\,s^2 + 0{,}7992\,s + 1}$	$1{,}9952 \times 10^6$

No.	Fungsi Alih	ISE
2	$G_H(s) = \frac{1{,}0202}{0{,}1449\ s^2 + 0{,}8315\ s + 1}$	$1{,}7162 \times 10^6$
3	$G_H(s) = \frac{1{,}0206}{0{,}1173\ s^2 + 0{,}8423\ s + 1}$	$1{,}8566 \times 10^6$
4	$G_H(s) = \frac{1{,}0211}{0{,}1676\ s^2 + \ 0{,}8777\ s + 1}$	$1{,}8306 \times 10^6$
5	$G_H(s) = \frac{1{,}0178}{0{,}1383\ s^2 + 0{,}7815\ s + 1}$	$1{,}8211 \times 10^6$

Berdasarkan fungsi alih dan perhitungan ISE yang ditunjukkan pada Tabel 3.4, dapat disimpulkan bahwa fungsi alih yang digunakan adalah fungsi alih nomor 2 dengan ISE paling kecil yaitu 1,7162 x 10^6. Fungsi alih sistem pada beban maksimal yaitu:

$$G_H(s) = \frac{1{,}0202}{0{,}1449\ s^2 + 0{,}8315\ s + 1} \tag{3.62}$$

3.2 Perancangan Kontroler Sistem

Perancangan kontroler yang pertama dilakukan adalah perancangan kontroler PID secara analitik dari parameter fungsi alih yang diperoleh dari hasil identifikasi. Setiap fungsi alih yang diperoleh dari kondisi masing-masing beban menghasilkan nilai Kp, Ki dan Kd. Hasil perancangan kontroler PID pada kondisi beban minimal, nominal dan maksimal, akan digunakan sebagai input untuk proses *learning NN* sampai mendapatkan bobot terbaik sehingga setelah proses *mapping NN* mampu melakukan penalaan terhadap kontroler PID yang memiliki keluaran berupa sinyal kontrol untuk mengatur kecepatan motor *spindle*.

3.2.1 Perancangan Kontroler PID

Perancangan kontroler PID diperlukan untuk mendapatkan nilai Kp, Ki dan Kd pada setiap kondisi pembebanan. Untuk mendapat parameter tersebut dilakukan perhitungan sebagai berikut:

1. Beban Minimal

   Perhitungan parameter PID untuk fungsi alih Persamaan 3.60 diperoleh:

$$K = 1{,}014 \tag{3.63}$$

$$\tau = 0{,}207 \tag{3.64}$$

$$\frac{1}{{\omega_n}^2} = 0{,}0101 \rightarrow \omega_n = \sqrt{\frac{1}{0.0101}} = 9{,}9504 \tag{3.65}$$

$$\frac{2\zeta}{\omega_n} = 0{,}2131 \rightarrow \zeta = \frac{0{,}2131\ .\ 9{,}9504}{2} = 1{,}0602 \tag{3.66}$$

   Untuk perancangan kontroler PID pada beban minimal, nilai τ yang diinginkan (τ^*) adalah 0,4 detik. Sehingga diperoleh parameter PID sebagai berikut:

$$\tau^* = 0{,}4 \tag{3.67}$$

$$Kp = \frac{2\zeta}{\tau^*\omega_n K} = \frac{2\ .\ 1{,}0602}{0{,}4\ .\ 9{,}9504\ .1{,}014} = 0{,}5254 \tag{3.68}$$

$$\tau_i = \frac{2\zeta}{\omega_n} = \frac{2\ .\ 1{,}0602}{9{,}9504} = 0{,}2131 \tag{3.69}$$

$$Ki = \frac{Kp}{\tau_i s} = 2{,}466 \tag{3.70}$$

$$\tau_d = \frac{1}{2\omega_n\zeta} = \frac{1}{2\ .\ 9{,}9504\ .1{,}0602} = 0{,}0473 \tag{3.71}$$

$$Kd = Kp\ .\tau_d s = 0{,}0249 \tag{3.72}$$

2. Beban Nominal

   Perhitungan parameter PID untuk fungsi alih Persamaan 3.61 diperoleh:

$$K = 1{,}014 \tag{3.73}$$

$$\tau = 0{,}426 \tag{3.75}$$

$$\frac{1}{{\omega_n}^2} = 0{,}0401 \rightarrow \omega_n = \sqrt{\frac{1}{0.0401}} = 4{,}9938 \tag{3.76}$$

$$\frac{2\zeta}{\omega_n} = 0{,}4208 \rightarrow \zeta = \frac{0{,}4208\ .\ 4{,}9938}{2} = 1{,}0507 \tag{3.77}$$

   Untuk perancangan kontroler PID pada beban nominal, nilai τ yang diinginkan (τ^*) adalah 0,4 detik. Sehingga diperoleh parameter PID sebagai berikut:

$$\tau^* = 0{,}4 \tag{3.78}$$

$$Kp = \frac{2\zeta}{\tau^* \omega_n K} = \frac{2\ .\ 1{,}0507}{0{,}4\ .\ 4{,}9938\ .1{,}014} = 1{,}0375 \tag{3.79}$$

$$\tau_i = \frac{2\zeta}{\omega_n} = \frac{2\ .\ 1{,}0507}{4{,}9938} = 0{,}4208 \tag{3.80}$$

$$Ki = \frac{Kp}{\tau_i s} = 2{,}4655 \tag{3.81}$$

$$\tau_d = \frac{1}{2\omega_n \zeta} = \frac{1}{2\ .\ 4{,}9938\ .1{,}0507} = 0{,}0953 \tag{3.82}$$

$$Kd = Kp\ .\ \tau_d s = 0{,}0989 \tag{3.83}$$

3. Beban Maksimal

Perhitungan parameter PID untuk fungsi alih Persamaan 3.61 diperoleh:

$$K = 1{,}0202 \tag{3.84}$$

$$\tau = 0{,}848 \tag{3.85}$$

$$\frac{1}{{\omega_n}^2} = 0{,}1449 \rightarrow \omega_n = \sqrt{\frac{1}{0.1449}} = 2{,}627 \tag{3.86}$$

$$\frac{2\zeta}{\omega_n} = 0{,}8315 \rightarrow \zeta = \frac{0{,}8315\ .\ 2{,}627}{2} = 1{,}0922 \tag{3.87}$$

Untuk perancangan kontroler PID pada beban maksimal, nilai τ yang diinginkan (τ^*) adalah 0,2 detik. Sehingga diperoleh parameter PID sebagai berikut:

$$\tau^* = 0{,}2 \tag{3.88}$$

$$Kp = \frac{2\zeta}{\tau^* \omega_n K} = \frac{2\ .\ 1{,}0922}{0{,}2\ .\ 2{,}627\ .1{,}0202} = 4{,}057 \tag{3.89}$$

$$\tau_i = \frac{2\zeta}{\omega_n} = \frac{2\ .\ 1{,}0922}{2{,}627} = 0{,}8315 \tag{3.90}$$

$$Ki = \frac{Kp}{\tau_i s} = 4{,}9006 \tag{3.91}$$

$$\tau_d = \frac{1}{2\omega_n \zeta} = \frac{1}{2\ .\ 2{,}627\ .1{,}0922} = 0{,}1743 \tag{3.92}$$

$$Kd = Kp\ .\ \tau_d s = 0{,}7101 \tag{3.93}$$

Hasil perancangan kontroler PID pada beban minimal, nominal dan maksimal diberikan pada Tabel 3.5.

Tabel 3.5. Parameter Kontroler PID Hasil Perhitungan

Parameter	Beban		
	Minimal	Nominal	Maksimal
K_p	0,5254	1,0375	4,075
K_i	2,466	2,4655	4,9006
K_d	0,0249	0,0989	0,7101

3.2.2 Perancangan Kontroler *Neural Network*

Setelah mendapatkan hasil parameter PID pada kondisi tiap-tiap, langkah selanjutnya adalah membangun struktur *NN* untuk proses *tuning* K_p, K_i dan K_d. Metode arsitektur *NN* yang digunakan adalah *back propagation.*

Untuk membuat struktur tuning *NN* dapat dilakukan dengan cara sebagai berikut:

1. Menentukan struktur kontroler
   Menetapkan jumlah layer input, layer *hidden* dan layer output. Lalu menetapkan bentuk fungsi aktivasi.
2. Menentukan model hasil desain yang diinginkan. Susun algoritma untuk *learning* bobot kontroler *NN* dengan *back propagation* sesuai persamaan berikut:
   -Formulasi *Forward*

$$Z_j^h \sum_{i,j=1}^{N} w_{ij}\, x_i \;;\; y_j^h = f_j(Z_j^h) \tag{3.94}$$

$$Z_p^o \sum_{k=1}^{m} w_{ko}\, y_k^h \;;\; y_p^o = f_p(Z_p^o) \tag{3.95}$$

   dengan fj dan fp adalah fungsi aktivasi.

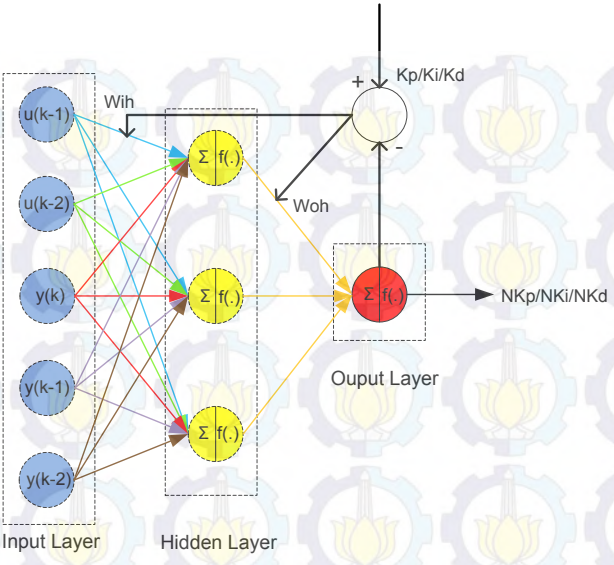
   -Formulasi *Back forward* (revisi bobot)
   Revisi bobot layer *hidden* ke output sesuai persamaan berikut:

$$w_o(k+1) = \; w_o(k) + \alpha.e.\lambda.Z_{hidden} \tag{3.96}$$

   Revisi bobot input ke *layer hidden* sesuai persamaan berikut:

$$w_{ih}(k+1) = \; w_{ih}(k) + \alpha.e.\lambda.X_i \tag{3.97}$$

Pada penelitian ini digunakan struktur seperti pada Gambar 3.5.



Gambar 3.5 Struktur *NN Back Propagation*

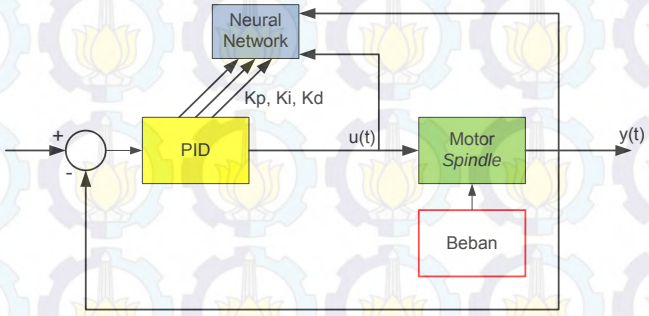
Gambar 3.5 menunjukkan strukur *NN back propagation* menggunakan 1 *layer* input dengan 5 unit masukan yaitu $u(k-1)$, $u(k-2)$, $y(k)$, $y(k-1)$ dan $y(k-2)$. Satu layer *hidden* dengan 3 unit *hidden* dan satu layer dengan satu unit output.

Prinsip kerja *NN* pada penelitian ini digunakan untuk mendapatkan nilai K_p, K_i dan K_d agar sesuai dengan nilai K_p, K_i dan K_d hasil perancangan. Jika hasil nilai PID pada *NN* belum sesuai, maka nilai bobot (w_{ih} dan w_{oh}) akan terus direvisi sampai mendapatkan nilai yang sesuai dengan nilai PID yang diinginkan.

3.2.3 Perancangan Proses *Learning NN*

Perancangan proses *learning*/belajar pada *Neural Network* ditunjukkan pada Gambar 3.6. Pada diagram blok sistem, masukan sistem $x(t)$ berupa kecepatan referensi yang diberikan sebesar 2000 rpm dan 3000 rpm yang berubah setiap 40 detik sekali. Blok PID berisi parameter nilai K_p, K_i dan K_d yang masing-masing mewakili parameter kontroler PID untuk beban minimal, nominal dan maksimal. Blok *Neural Network* berisi struktur *NN* yang berisi algoritma untuk

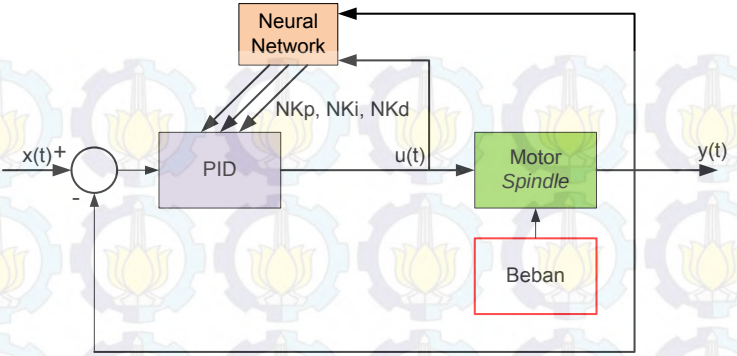
melakukan proses *learning NN*. Input untuk *NN* adalah parameter PID yang terdiri dari K_p, K_i dan K_d, sinyal kontrol yang terdiri dari $u(k-1)$, $u(k-2)$ dan sinyal output yang terdiri dari $y(k)$, $y(k-1)$ dan $y(k-2)$. Blok motor *spindle* merupakan subsistem dari model identifikasi *plant*. Blok beban merupakan kondisi pembebanan yang diberikan pada motor *spindle,* perubahan beban yang diberikan yaitu [1 2 2 1 1 3 3 1] yang berubah setiap 10 detik sekali. Dengan 1 adalah mempresentasikan beban minimal, 2 adalah mempresentasikan beban nominal dan 3 adalah mempresentasikan beban maksimal. $Y(t)$ adalah keluaran sistem berupa kecepatan motor *spindle* dalam rpm.



Gambar 3.6 Diagram Blok Proses *Learning Neural Network*

3.2.4 Perancangan Proses *Mapping NN*

Setelah proses belajar pada *NN* telah mampu mencapai nilai yang mendekati target, maka proses belajar pada *NN* pun selesai dan selanjutnya melakukan proses *mapping*. Proses *mapping* adalah proses pemilihan bobot terbaik untuk mendapatkan *error* terkecil dalam penentuan nilai target. Pada proses *mapping* pula, revisi bobot tidak lagi dilakukan karena algoritma telah diganti dengan memasukan nilai bobot yang telah didapat dari proses *learning*. Dengan proses *mapping* pula menunjukkan pula bahwa kombinasi antara kontroler PID dengan mekanisme tuning *NN* telah dapat dilakukan untuk *plant* dengan kondisi perubahan tertentu. Kombinasi kontroler tersebut disebut *Neuro-PID*. Kontroler *Neuro-PID* pada plan motor s*pindle* secara keseluruhan ditunjukkan pada Gambar 3.7.



Gambar 3.7 Diagram Blok Proses *Mapping Neural Network*

Pada diagram blok sistem, masukan sistem $x(t)$ berupa kecepatan referensi yang diberikan sebesar 2000 rpm dan 3000 rpm yang berubah setiap 40 detik sekali. Blok *Neural Network* berisi struktur *NN* yang berisi algoritma untuk melakukan proses *mapping NN*. Input untuk *NN* adalah sinyal kontrol yang terdiri dari $u(k-1)$, $u(k-2)$ dan sinyal output yang terdiri dari $y(k)$, $y(k-1)$ dan $y(k-2)$. Output dari *NN* masuk ke dalam blok PID untuk memberikan nilai parameter NK_p, NK_i dan NK_d sebagai hasil dari algoritma proses *mapping*. Sinyal kontrol dari *Neuro-PID* masuk ke dalam blok motor *spindle* untuk mengatur kecepatan sesuai dengan perubahan beban yang terjadi.

Halaman ini sengaja dikosongkan

BAB IV
PENGUJIAN DAN ANALISIS

Bab ini akan membahas pengujian dan analisa dari simulasi perancangan sistem. Simulasi dilakukan dengan menggunakan *software* MATLAB. Pengujian dan analisa yang dilakukan diantaranya adalah respon sistem tanpa kontroler dari setiap kondisi beban dengan kecepatan referensi konstan, respon dari setiap sistem dengan kontroler PID, respon sistem hasil *learning Neural Network* dari setiap nilai parameter K_p, K_i dan K_d, respon sistem dengan kontroler *Neuro-PID* pada kondisi perubahan beban tertentu dan kecepatan referensi konstan, dan respon sistem dengan kontroler *Neuro-PID* pada kondisi pembebanan tertentu dan kecepatan referensi berubah.

4.1 Pengujian Sistem Tanpa Kontroler

Pengujian sistem tanpa kontroler yaitu menguji sistem secara lup terbuka untuk mengetahui respon *plant* saat diberi input berupa kecepatan referensi. Dari fungsi alih yang didapat pada bab 3 untuk kondisi beban minimal, nominal dan maksimal, dapat dibuat simulasinya pada *Simulink* MATLAB dengan terlebih dahulu melakukan perhitungan utntuk masing-masing fungsi alih beban seperti pada persamaan *state space* sebagai berikut:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K}{as^2+bs+1} \tag{4.1}$$

$$a\ddot{y} + b\dot{y} = ku \tag{4.2}$$

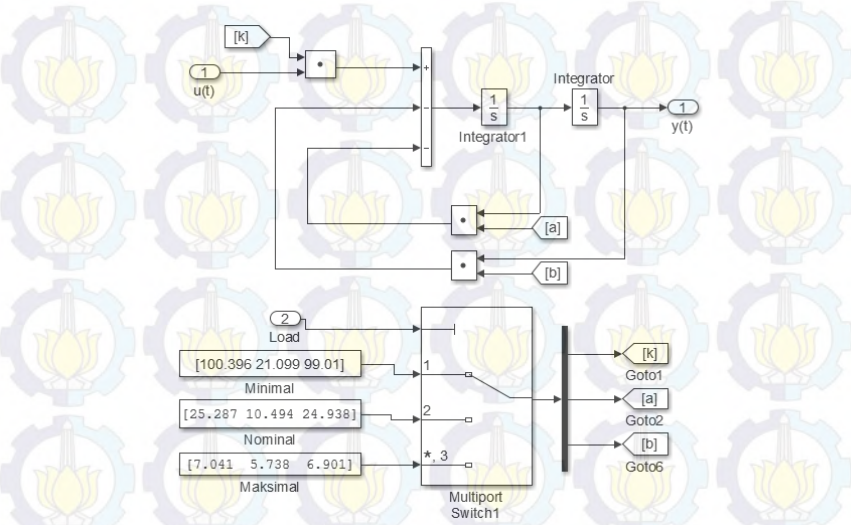
$$\ddot{y} = -a\dot{y} - by + ku \tag{4.3}$$

Hasil perhitungan dari Persamaan 4.1, 4.2 dan 4.3 ditunjukkan pada Tabel 4.1.

Tabel 4.1 Hasil Perhitungan Persamaan *State Space*

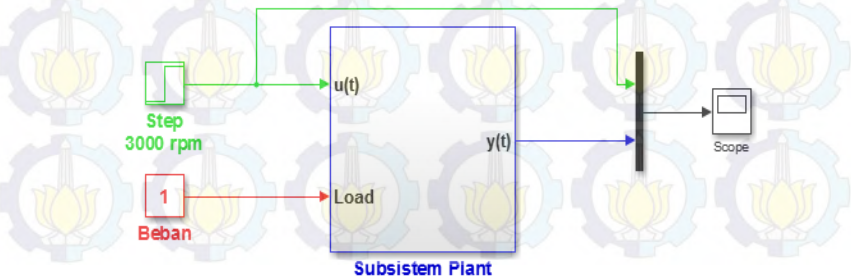
Fungsi Alih	k	a	b
Minimal	100,396	21,099	99,01
Nominal	25,287	10,494	24,938
Maksimal	7,041	5,738	6,901

Dari hasil perhitungan pada Tabel 4.1, dapat dibuat simulasi subsistem pada *Simulink* seperti Gambar 4.1.



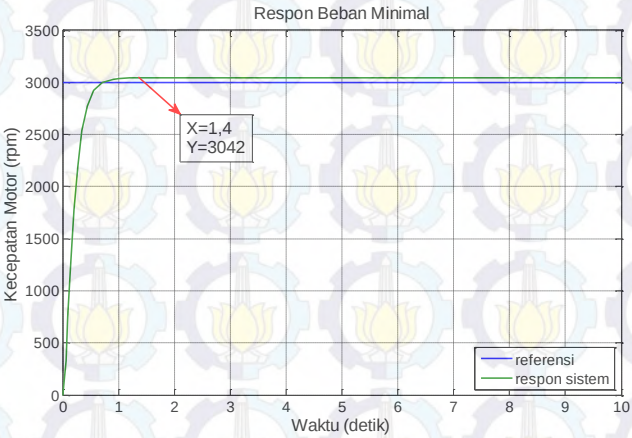
Gambar 4.1 Skema Subsistem *Plant*

Setelah membuat simulasi subsistem, selanjutnya melakukan pengujian pada rangkaian lup terbuka dengan memberi masukan sinyal step sebagai representasi dari kecepatan referensi sebesar 3000 rpm. Untuk kondisi pembebanan dilakukan dengan memberikan nilai konstanta pada blok beban. Nilai 1 untuk beban minimal, nilai 2 untuk beban nominal dan nilai 3 untuk beban maksimal. Skema pengujian tiap beban tanpa kontroler ditunjukkan pada Gambar 4.2.



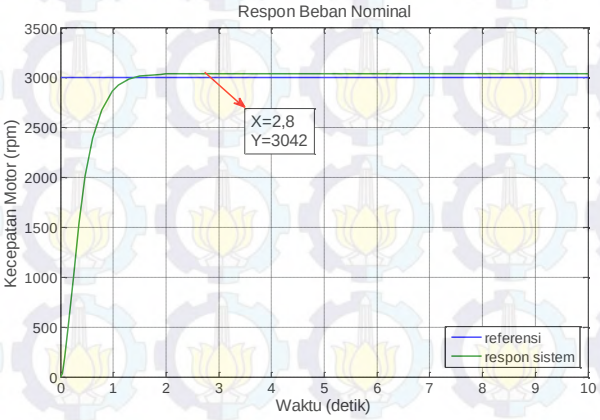
Gambar 4.2 Skema Pengujian Tiap Beban

Respon dari hasil pengujian sistem tanpa kontroler dapat dilihat pada Gambar 4.3, 4.4 dan 4.5



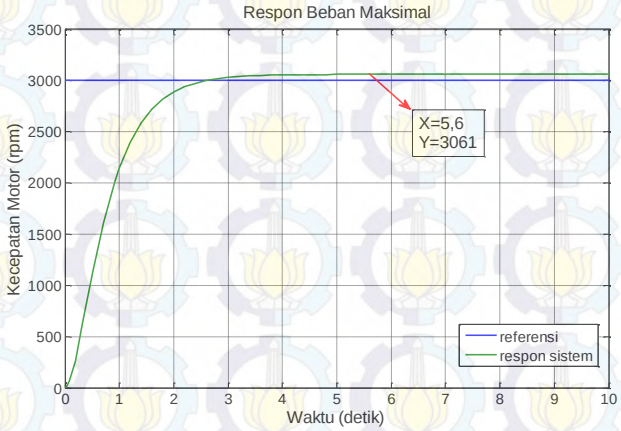
Gambar 4.3 Respon Sistem pada Beban Minimal

Pada Gambar 4.3 terlihat bahwa keluaran *plant* tidak sesuai dengan masukan nilai referensi karena masih terdapat *error steady state* sebesar 42 rpm atau setara dengan 1,4% dari kecepatan yang diinginkan yaitu 3000 rpm.



Gambar 4.4 Respon Sistem pada Beban Nominal

Pada Gambar 4.4 terlihat bahwa keluaran *plant* tidak sesuai dengan masukan nilai referensi karena masih terdapat *error steady state* sebesar 42 rpm atau setara dengan 1,4% dari kecepatan yang diinginkan yaitu 3000 rpm.



Gambar 4.5 Respon Sistem pada Beban Maksimal

Pada Gambar 4.5 terlihat bahwa keluaran *plant* tidak sesuai dengan masukan nilai referensi karena masih terdapat *error steady state* sebesar 61 rpm atau setara dengan 2,03% dari kecepatan yang diinginkan yaitu 3000 rpm.

Hasil respon dari masing-masing kondisi pembebanan tanpa kontroler ditunjukkan pada Tabel 4.1.

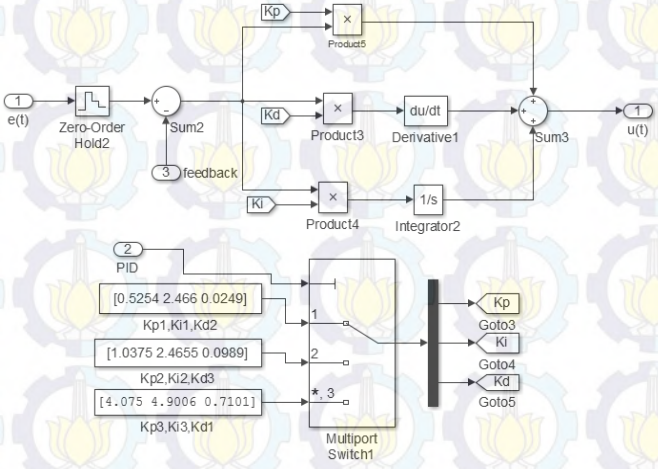
Tabel 4.2 Hasil Respon Sistem Tanpa Kontroler

Spesifikasi	Beban Minimal	Beban Nominal	Beban Maksimal
τ	0,22 detik	0,44 detik	0,86 detik
t_S (5%)	0,66 detik	1,32 detik	2,58 detik
t_r (5%-95%)	0,46 detik	0,7 detik	1,76 detik
td	0,15 detik	0,3 detik	0,6 detik
Ess	1,4%	1,4%	2,03%

Berdasarkan pengujian *plant* tanpa kontroler pada tiap-tiap kondisi pembebanan, ternyata tidak sesuai dengan masukan nilai referensi yang diinginkan. Sehingga perlu perlu diberikan kontroler agar *plant* dapat mencapai repon sistem yang dinginkan.

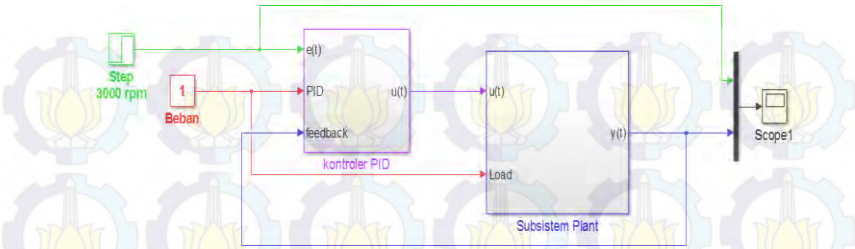
4.2 Pengujian Kontroler PID pada Tiap Beban

Perancangan kontroler PID pada bab 3 telah menghasilkan parameter nilai K_p, K_i dan K_d untuk masing-masing kondisi pembebanan. Selanjutnya untuk melakukan pengujian sistem dengan kontroler, perlu dibuat simulasi kontroler PID pada *Simulink* seperti yang ditunjukkan pada Gambar 4.6.

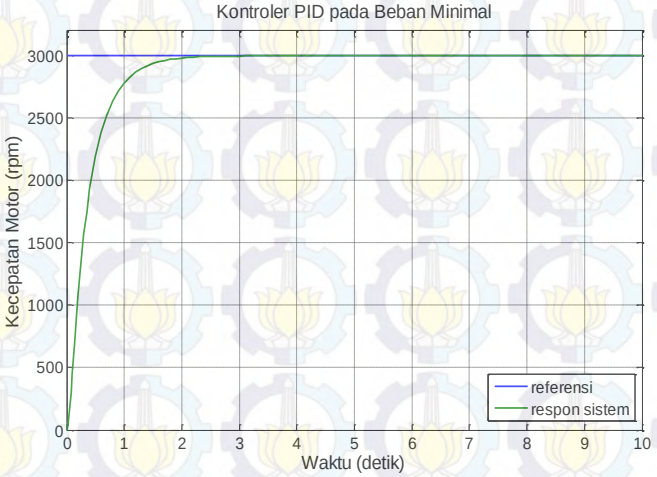


Gambar 4.6 Skema Kontroler PID

Setelah membuat simulasi kontroler PID selanjutnya melakukan pengujian pada rangkaian lup tertutup dengan memberi masukan sinyal step sebagai representasi dari kecepatan referensi sebesar 3000 rpm. Untuk kondisi pembebanan dilakukan dengan memberikan nilai konstanta pada blok beban. Nilai 1 untuk beban minimal, nilai 2 untuk beban nominal dan nilai 3 untuk beban maksimal. Skema pengujian tiap beban dengan kontroler ditunjukkan pada Gambar 4.7.

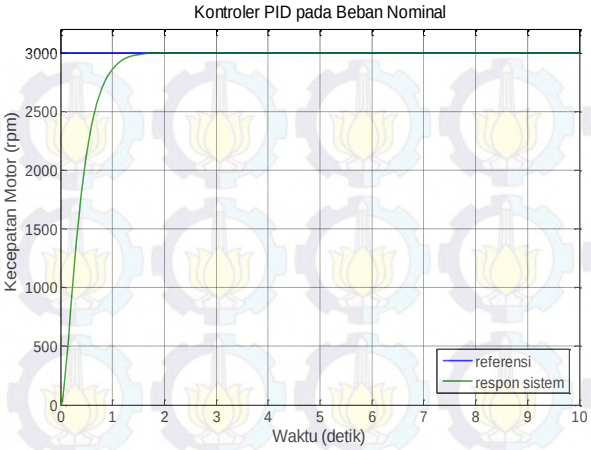


Gambar 4.7 Skema Pengujian Kontroler PID pada Tiap Beban



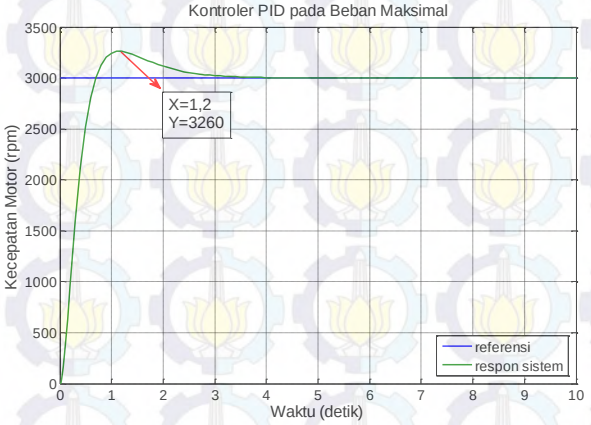
Gambar 4.8 Respon Kontroler PID pada Beban Minimal

Pada Gambar 4.8 terlihat bahwa respon sistem mampu mengikuti input kecepatan referensi yang diberikan sebesar 3000 rpm setelah 2 detik.



Gambar 4.9 Respon Kontroler PID pada Beban Nominal

Pada Gambar 4.9 terlihat bahwa respon sistem mampu mengikuti input kecepatan referensi yang diberikan sebesar 3000 rpm setelah 1,5 detik.



Gambar 4.10 Respon Kontroler PID pada Beban Maksimal

Pada Gambar 4.10 terlihat bahwa respon sistem mampu mengikuti input kecepatan referensi yang diberikan sebesar 3000 rpm

setelah 3 detik. Namun sempat terjadi *overshoot* saat 1,2 detik sebesar 3260 rpm atau setara dengan 8,67% dari kecepatan referensi yang diinginkan.

Tabel 4.3 Hasil Respon Sistem dengan Kontroler PID

Spesifikasi	Beban Minimal	Beban Nominal	Beban Maksimal
τ	0,39 detik	0,43 detik	0,36 detik
t_S (5%)	1,17 detik	1,29 detik	1,08 detik
t_r (5%-95%)	1,104 detik	0,9 detik	0,56 detik
t_d	0,27 detik	0,3 detik	0,25 detik
Ess	0%	0%	8,67%

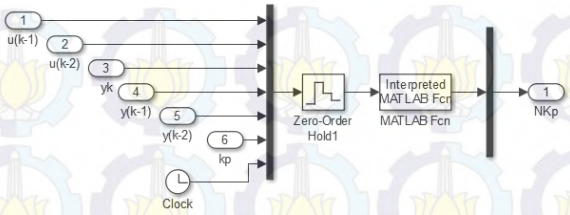
4.3 Pengujian Proses *Learning NN* Kontroler

Agar sistem dapat beradaptasi dengan perubahan kondisi beban, maka perlu mekanisme penalaan kontroler PID yang dilakukan oleh *Neural Network*. Pada *NN* diperlukan proses *learning* untuk mendapatkan nilai bobot terbaik sehingga kontroler nantinya dapat menentukan nilai K_p, K_i dan K_d secara otomatis sesuai kondisi pembebanan yang diberikan.

Untuk simulasi proses *learning NN* pada *Simulink* dibuat masukan berupa $u(k-1)$ dan $u(k-2)$ yang merupakan sinyal kontrol dalam bentuk diskrit, $y(k)$, $y(k-1)$ dan $y(k-2)$ yang merupakan sinyal keluaran dalam bentuk diskrit dan K_p sebagai nilai target yang ingin dicapai sekaligus digunakan untuk mencari selisih *error* antara output dan K_p sehingga revisi bobot pada *NN* dapat dilakukan. Pada blok *Matlab Function*, dibuat program untuk proses *learning NN* dengan arsitektur *back propagation*. Parameter yang diberikan pada program tersebut meliputi:

- Jumlah unit input = 5
- Jumlah unit *hidden layer* = 3
- Inisialisasi bobot input *hidden* (w_{ih})= ones
- Inisialisasi bobot output *hidden* (w_{oh}) = ones
- Fungsi aktivasi linear λ= 1
- *Learning rate* untuk bobot input ke *hidden layer* = 0,001

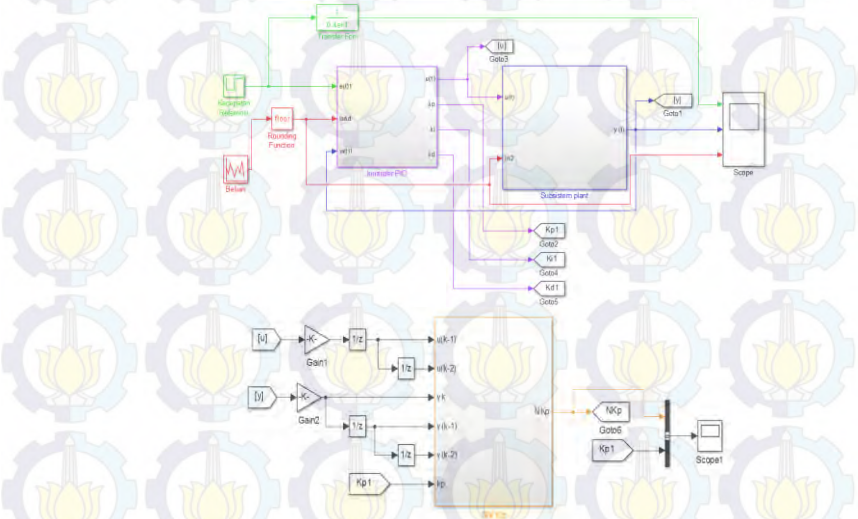
- *Learning rate* untuk bobot *hidden layer* ke output = 0,001



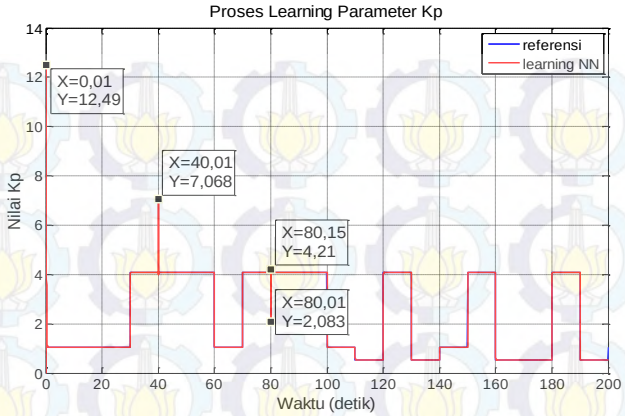
Gambar 4.11 Skema Proses *learning NN* untuk Parameter K_p

Untuk proses *learning NN* pada parameter K_i dan K_d juga dilakukan seperti pada parameter K_p.

Tahap selanjutnya yaitu pengujian dengan memberikan masukan berupa sinyal *repeating sequence stair* yang merupakan representasi dari perubahan kecepatan referensi motor 2000 rpm dan 3000 rpm setiap 40 detik dan sinyal *random* yang merupakan representasi dari perubahan kondisi beban. Skema pengujian proses *learning NN* ditunjukkan pada Gambar 4.12.

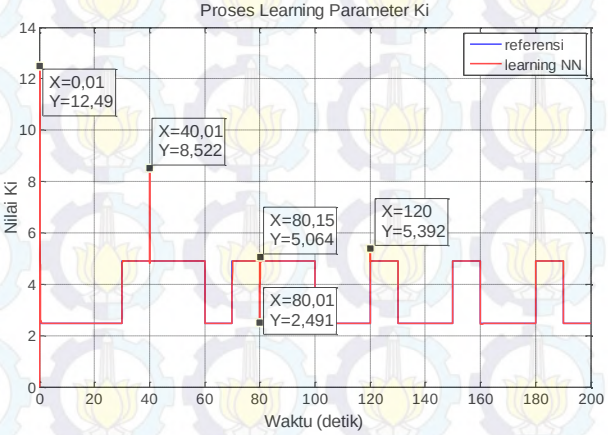


Gambar 4.12 Skema Pengujian Proses *learning NN* untuk Parameter K_p



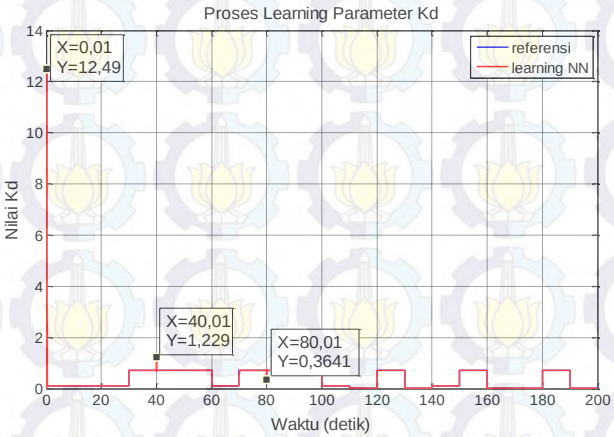
Gambar 4.13 Proses *Learning NN* untuk Parameter K_p

Pada Gambar 4.13 terlihat bahwa *NN* mampu belajar mengikuti parameter nilai K_p yang berubah menyesuaikan kondisi beban. Walaupun terjadi *error* di detik ke-0.01, 40 dan 80 yang disebabkan oleh revisi bobot yang dilakukan selama proses *learning* namun *NN* mampu dengan cepat mengikuti perubahan nilai parameter K_p yang diberikan.



Gambar 4.14 Proses *Learning NN* untuk Parameter K_i

Pada Gambar 4.14 terlihat bahwa *NN* mampu belajar mengikuti parameter nilai K_i yang berubah menyesuaikan kondisi beban. Walaupun terjadi *error* di detik ke-0.01, 40 dan 80 yang disebabkan oleh revisi bobot yang dilakukan selama proses *learning* namun *NN* mampu dengan cepat mengikuti perubahan nilai parameter K_i yang diberikan.

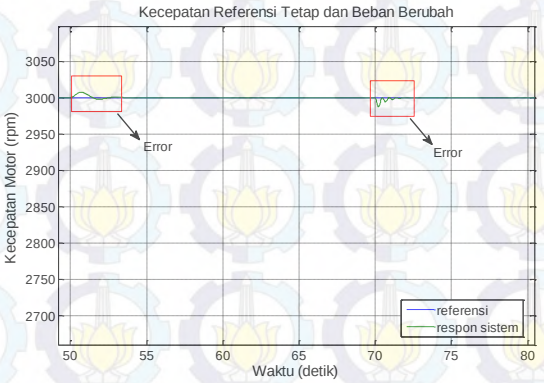


Gambar 4.15 Proses *Learning NN* untuk Parameter K_d

Pada Gambar 4.15 terlihat bahwa *NN* mampu belajar mengikuti parameter nilai K_d yang berubah menyesuaikan kondisi beban. Walaupun terjadi *error* di detik ke-0.01, 40 dan 80 yang disebabkan oleh revisi bobot yang dilakukan selama proses *learning* namun *NN* mampu dengan cepat mengikuti perubahan nilai parameter K_d yang diberikan.

4.4 Pengujian Kontroler *Neuro-PID*

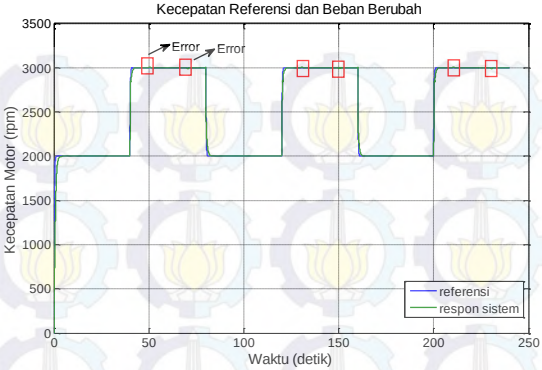
Setelah bobot terbaik didapat, lalu melakukan proses *mapping* pada algoritma program JST. Dari hasil *mapping*, sistem diuji dengan memberikan input berupa sinyal step yang mempresentasikan kecepatan referensi tetap dan sinyal *repeating sequence stair* yang mempresentasikan kondisi perubahan beban setiap 10 detik dengan pola [1 2 2 1 1 3 3 1]. Dengan nilai 1 untuk mbeban minimal, nilai 2 untuk beban nominal dan nilai 3 untuk beban maksimal. Respon sistem ditunjukkan pada Gambar 4.16.



Gambar 4.16 Respon Sistem Saat Kecepatan Referensi Tetap dan Beban Berubah

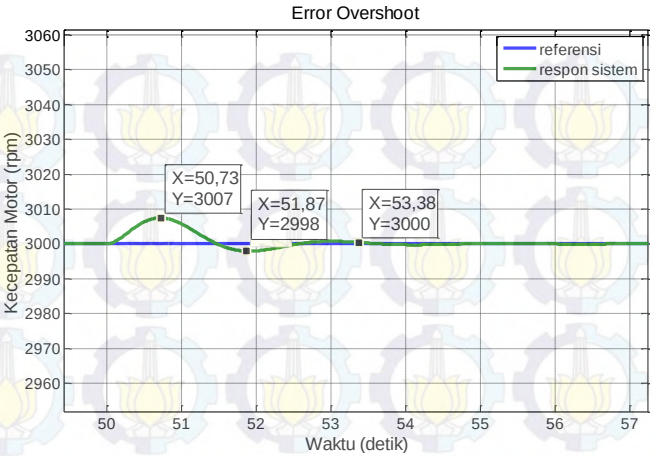
Pada Gambar 4.16 terlihat bahwa secara keseluruhan, kontroler mampu beradaptasi saat diberi kecepatan referensi tetap dengan kondisi beban yang berubah. *Error overshoot* yang terjadi saat detik ke-50 adalah adanya perubahan beban dari minimal ke beban maksimal. Pada detik ke-70 terdapat osilasi yang terjadi karena adanya perubahan beban dari maksimal ke minimal. Nilai *error overshoot* dan osilasi yang terjadi masih dalam batas toleransi karena nilainya yang relatif kecil.

Pengujian selanjutnya yaitu dengan memberikan input berupa sinyal *repeating sequence stair* yang merupakan representasi dari perubahan kecepatan referensi motor dari 2000 rpm dan 3000 rpm setiap 40 detik dan memberikan perubahan beban sama dengn pola seperti pengujian sebelumnya. Gambar 4.17 merupakan hasil respon sistem.



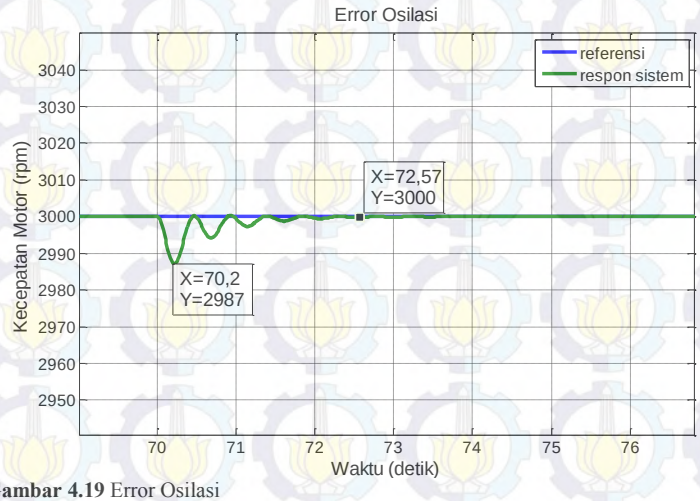
Gambar 4.17 Respon Sistem Saat Kecepatan Referensi dan Beban Berubah

Pada Gambar 4.17 simulasi yang dijalankan selama 240 detik ini, terlihat bahwa secara keseluruhan, kontroler mampu beradaptasi dengan kondisi beban yang berubah. *Error overshoot* yang terjadi saat detik ke-50, ke-130 dan ke-210 disebabkan karena adanya perubahan dari beban minimal ke beban maksimal. Sedangkan osilasi yang terjadi pada detik ke-70, ke-150 dan ke-230 disebabkan karena adanya perubahan dari beban maksimal ke beban minimal. Nilai *error overshoot* dan osilasi yang terjadi masih dalam batas toleransi karena nilainya yang relatif kecil. Nilai *error overshoot* dan osilasi lebih jelasnya ditunjukkan pada Gambar 4.18 dan 4.19.



Gambar 4.18 Besar Nilai *Overshoot*

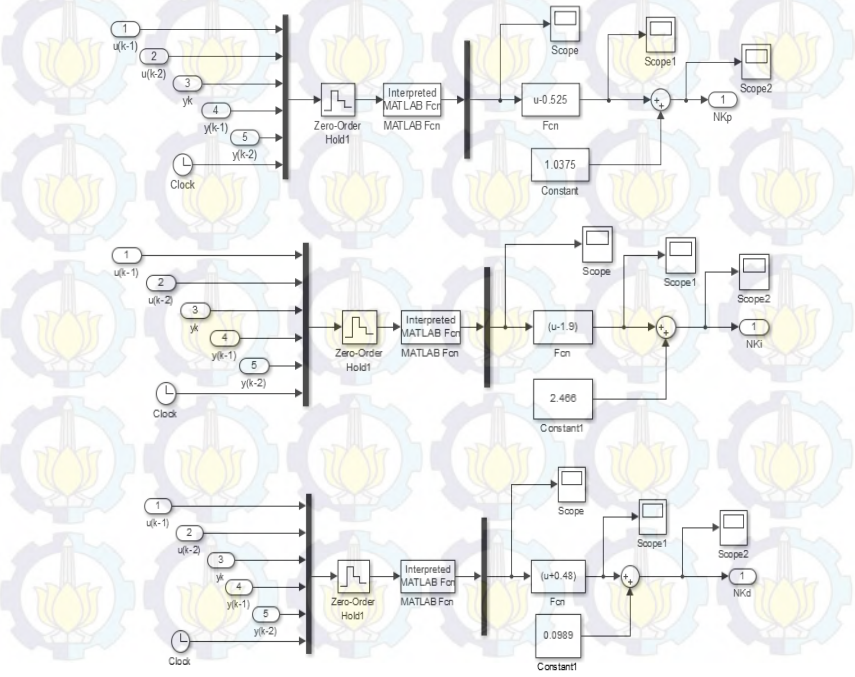
Pada Gambar 4.16 dapat terlihat bahwa besar *overshoot* yang terjadi saat perubahan beban minimal ke maksimal yaitu sebesar 7 rpm atau setara dengan 0,23% dari kecepatan referensi yaitu 3000 rpm.



Gambar 4.19 Error Osilasi

Osilasi terjadi di setiap perubahan beban maksimal ke maksimal. Pada Gambar 4.19 terlihat besarnya yang osilasi terjadi yaitu 2987 rpm dan respon sistem membutuhkan waktu selama 2,57 detik untuk mencapai nilai referensi dan mencapai kondisi stabil.

Overshoot dan osilasi terjadi karena saat merancang kontroler PID untuk kondisi beban maksimal, nilai parameter K_p, K_i dan K_d belum bisa menghasilkan respon sistem yang diinginkan. Respon sistem pada beban maksimal dengan kontroler PID masih terdapat *overshoot* sebesar 3260 rpm. Untuk mendapatkan respon yang diinginkan, kontroler PID beban maksimal tidak dilakukan *tuning* manual, namun modifikasi dilakukan pada output dari kontroler *Neuro-PID*. Modifikasi pada output kontroler dilakukan dengan cara memberikan nilai konstanta parameter PID beban nominal pada masing-masing blok *simulink* NK_p, NK_i dan NK_d, yang ditunjukkan pada Gambar 4.20.



Gambar 4.20 Modifikasi pada Output Kontroler

Hasil respon sistem dari kontroler *Neuro-PID* yang telah dimodifikasi pada bagian output, telah menunjukkan respon sistem yang diinginkan dan dapat beradaptasi dengan kondisi perubahan tertentu seperti yang telah terlihat pada Gambar 4.17.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa, maka diperoleh kesimpulan sebagai berikut:

1) Hasil perancangan PID dengan nilai K_p, K_i dan K_d pada tiap-tiap kondisi beban telah mampu mengatur kecepatan motor *spindle* mengikuti kecepatan referensi yang diberikan. *Neural network* mampu mengikuti nilai parameter K_p, K_i dan K_d yang diberikan saat proses l*earning*. *Neural Network* mampu beradaptasi dengan kondisi perubahan beban dan memperbaiki *overshoot* yang terjadi oleh parameter PID saat beban maksimal, tanpa memerlukan tuning manual. Modifikasi pada output *neural network* masih diperlukan agar sistem dapat mengikuti respon transien yang diinginkan.
2) Kontroler *Neuro-PID* mampu mengatur kestabilan kecepatan motor *spindle* dengan kecepatan referensi berubah dari 2000 rpm ke 3000 rpm dan pada kondisi beban dengan perubahan tertentu. Hal ini ditunjukkan dengan kecilnya nilai *error overshoot* yang hanya terjadi saat perubahan beban minimal ke maksimal di kecepatan 3000 rpm yaitu sebesar 0,23%.
3) Dengan menggunakan kontroler *Neuro-PID, plant* dapat beradaptasi terhadap perubahan beban dengan baik dibandingkan dengan hanya menggunakan kontroler PID yang masih mengalami *overshoot* sebesar 8,67% saat kondisi beban maksimal.

5.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya adalah perlunya melakukan banyak percobaan pada proses *learning neural network* agar benar-benar mendapatkan bobot terbaik sehingga pada output kontroler tidak perlu lagi melakukan modifikasi untuk mendapatkan respon sistem yang diinginkan. Dan diharapkan kedepannya Tugas Akhir ini dapat dijadikan sebagai acuan agar Untuk penelitian selanjutnya Tugas Akhir ini dapat dijadikan sebagai acuan agar penelitian selanjutnya dapat merancang kontroler yang lebih baik lagi yang dapat tahan terhadap perubahan beban yang lebih bervariasi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Fitria Kusuma Kartini, Adelina. "Sistem Pengaturan Gerakan *Tool* pada Prototipe Mesin CNC dengan Kontroler *Distrubance Observer*", Tugas Akhir Teknik Elektro-ITS.2012.

[2] Suryaditya Putri, Nindita. "Pengaturan Proses *Face Milling* pada Mesin *Computer Numerical Control* (CNC) dengan Kontroler *Fuzzy-PID"*. Tugas Akhir-Teknik Elektro ITS. 2013.

[3] Setyaningrum, Dinar. "Desain dan Implementasi *Model Reference Adaptive Control* untuk Pengaturan *Tracking Optimal* Posisi Motor DC". Tugas Akhir-Teknik Elektro ITS. 2012.

[4] Gamayanti, Nurlita. "Karakteristik Orde Pertama". *Handout* Mata Kuliah Dasar Sistem Pengaturan, Jurusan Teknik Elektro ITS.

[5] Pramudijanto, Josaphat. "Identifikasi Model Matematik". *Handout* Mata Kuliah Dasar Sistem Pengaturan, Jurusan Teknik Elektro ITS.

[6] Jakoubek, Ing. Pavel. "*Experimental Identification of Stabile Nonoscillatory Systems from Step-Reponses by Selected Methods*". Konference *Studentské tvůrčí činnosti*, 2009.

[7] Drs. Jong Jek Siang, M. "Jaringan Syaraf Tiruan & Pemrogramannya menggunakan MATLAB". Yogyakarta: Penerbit ANDI. 2004

Halaman ini sengaja dikosongkan

LAMPIRAN

1. Program *Learning Neural Network* untuk Parameter Kp

```
function Kp=kp1(in1)
global tt jin1 jeh1   lmdh lmdo alph1 alpo1 wih1
woh1  kp1 biasKp
%xx  = in1(1:5);
kp1  = in1(6);
tt = in1(7);
% proses inisialisasi
if tt==0
    jin1 = 5; %Jumlah unit input
    jeh1 = 3; %Jumlah unit HL.
    wih1 = ones(jin1,jeh1);%bobot input hidden
    woh1 = ones(jeh1,1);%bobot output hidden
    lmdh=1;%fungsi aktivasi hidden layer
    lmdo=1;%fungsi aktivasi output layer
    alph1=0.001;%learning rate hidden
    alpo1=0.001;%learning rate output
  % biasKp=1.0375;
end
    for x=1:jin1
        xx(x)=in1(x);
    end
% Perhitungan forward
% Menghitung output layer hidden
    for j=1:jeh1
        zh1(j)=0;%z hidden
        for i=1:jin1
            zh1(j)=zh1(j)+wih1(i,j)*xx(i)%Jumlah
keluaran hidden layer
        end
yh1(j)=lmdh*zh1(j)  % keluaran hidden= Jumlah HL
dikali fungsi aktivasi linear
    end
    % menghitung output neuron
    zo1=0;%z output
    for j=1:jeh1
        zo1=zo1+woh1(j)*yh1(j);
```

```
    end
%zo1=zo1+biasKp
    yn1=lmdo*zo1; % fungsi aktivasi linear
                  % yn1=NKp
% menghitung eror output
    er1=kp1-yn1;% perhitungan backward (revisi
bobot)
% revisi bobot dari layer hidden ke layer output
    for j=1:jeh1
        woh1(j)=woh1(j)+alpo1*er1*lmdo*yh1(j);
    end
% revisi bias
% biasKp=biasKp+alpo1*er1;
% menghitung perambatan eror
    for j=1:jeh1
        if zo1==0
            erh1(j)=er1/jeh1;
        else
            erh1(j)=(woh1(j)*yh1(j)/zo1)*er1;
        end
    end
% revisi bobot dari layer input ke layer hidden
    for j=1:jeh1
        for i=1:jin1

wih1(i,j)=wih1(i,j)+alph1*erh1(j)*lmdh*xx(i);
        end
    end
    assignin('base','wih_p',wih1);
    assignin('base','woh_p',woh1);
Kp=yn1;
```

2. Program *Learning Neural Network* untuk Parameter Ki

```
function Ki=jstKi(in2)
global tt jin2 jeh2 lmdh2 lmdo2 alph2 alpo2 wih2
woh2  ki1
%xx  = in2(1:5);
ki1  = in2(6);
tt = in2(7);
```

```
% proses inisialisasi
if tt==0
    jin2 = 5; %Jumlah unit input
    jeh2 = 3; %Jumlah unit HL.
    wih2 = ones(jin2,jeh2);%bobot input hidden
    woh2 = ones(jeh2,1);%bobot output hidden
    lmdh2=1;%fungsi aktivasi hidden layer
    lmdo2=1;%fungsi aktivasi output layer
    alph2=0.001;%learning rate hidden
    alpo2=0.001;%learning rate output
   % biasKp=1.0375;
end
    for x=1:jin2
        xx(x)=in2(x);
    end
    % Perhitungan forward
    % Menghitung output layer hidden
    for j=1:jeh2
        zh2(j)=0;%z hidden
        for i=1:jin2
            zh2(j)=zh2(j)+wih2(i,j)*xx(i)%Jumlah
keluaran hidden layer
        end
        yh2(j)=lmdh2*zh2(j) % keluaran hidden=
Jumlah HL dikali fungsi aktivasi linear
    end
    % menghitung output neuron
    zo2=0;%z output
    for j=1:jeh2
        zo2=zo2+woh2(j)*yh2(j);
    end
    %zo1=zo1+biasKp
    yn2=lmdo2*zo2; % fungsi aktivasi linear
                   % yn1=NKp
    % menghitung eror output
    er2=ki1-yn2;
% perhitungan backward (revisi bobot)
% revisi bobot dari layer hidden ke layer output
    for j=1:jeh2
        woh2(j)=woh2(j)+alpo2*er2*lmdo2*yh2(j);
```

```
    end
% revisi bias
% biasKp=biasKp+alpo1*er1;
% menghitung perambatan eror
    for j=1:jeh2
        if zo2==0
            erh2(j)=er2/jeh2;
        else
            erh2(j)=(woh2(j)*yh2(j)/zo2)*er2;
        end
    end
% revisi bobot dari layer input ke layer hidden
    for j=1:jeh2
        for i=1:jin2
wih2(i,j)=wih2(i,j)+alph2*erh2(j)*lmdh2*xx(i);
        end
    end
    assignin('base','wih_i',wih2);
    assignin('base','woh_i',woh2);
Ki=yn2;
```

3. Program *Learning Neural Network* untuk Parameter Kd

```
function Kd=jstKd(in3)
global tt jin3 jeh3 lmdh3 lmdo3 alph3 alpo3 wih3
woh3  kd1
kd1  = in3(6);
tt = in3(7);
% proses inisialisasi
if tt==0
    jin3 = 5; %Jumlah unit input
    jeh3 = 3; %Jumlah unit HL
    wih3 = ones(jin3,jeh3);%bobot input hidden
    woh3 = ones(jeh3,1);%bobot output hidden
    lmdh3=1;%fungsi aktivasi hidden layer
    lmdo3=1;%fungsi aktivasi output layer
    alph3=0.001;%learning rate hidden
    alpo3=0.001;%learning rate output

end
```

```
    for x=1:jin3
        xx(x)=in3(x);
    end
    % Perhitungan forward
    % Menghitung output layer hidden
    for j=1:jeh3
        zh3(j)=0;%z hidden
        for i=1:jin3
            zh3(j)=zh3(j)+wih3(i,j)*xx(i)%Jumlah
keluaran hidden layer
        end
        yh3(j)=lmdh3*zh3(j) % keluaran hidden=
Jumlah HL dikali fungsi aktivasi linear
    end
    % menghitung output neuron
    zo3=0;%z output
    for j=1:jeh3
        zo3=zo3+woh3(j)*yh3(j);
    end
    %zo1=zo1+biasKp
    yn3=lmdo3*zo3; % fungsi aktivasi linear
                   % yn1=NKp
    % menghitung eror output
    er3=kd1-yn3;
% perhitungan backward (revisi bobot)
% revisi bobot dari layer hidden ke layer output
    for j=1:jeh3
        woh3(j)=woh3(j)+alpo3*er3*lmdo3*yh3(j);
    end
    %revisi bias
  %  biasKp=biasKp+alpo1*er1;

    % menghitung perambatan eror
    for j=1:jeh3
        if zo3==0
            erh3(j)=er3/jeh3;
        else
            erh3(j)=(woh3(j)*yh3(j)/zo3)*er3;
        end
    end
```

```
 % revisi bobot dari layer input ke layer hidden
    for j=1:jeh3
        for i=1:jin3

wih3(i,j)=wih3(i,j)+alph3*erh3(j)*lmdh3*xx(i);
        end
    end
    assignin('base','wih_d',wih3);
    assignin('base','woh_d',woh3);
Kd=yn3;
```

4. Program *Mapping Neural Network* untuk Parameter Kp

```
function MKp=mappingKp(in11)
global tt jin11 jeh11   lmdh lmdo  wih11 woh11
biaskp1
tt = in11(6);
% proses inisialisasi

if tt==0
    jin11 = 5; %Jumlah unit input
    jeh11 = 3; %Jumlah unit HL.
    wih11 =
[0.972213477255337,0.972213477255337,
0.972213477255337;
0.973510443320408,0.973510443320408,
0.973510443320408;0.963235949240228,
0.963235949240228,0.963235949240228;
0.963751092633304,0.963751092633304,
0.963751092633304;0.964282153872162,
0.964282153872162,0.964282153872162];%bobot
input hidden
    woh11 = [-0.000852367504637727
             -0.000852367504637727
             -0.000852367504637727];%bobot
output hidden

    lmdh=1;%fungsi aktivasi hidden layer
    lmdo=1;%fungsi aktivasi output layer
end
```

```
    for x=1:jin11
        xx1(x)=in11(x);
    end
    % Perhitungan forward
    % Menghitung output layer hidden
    for j=1:jeh11
        zh11(j)=0;%z hidden
        for i=1:jin11

zh11(j)=zh11(j)+wih11(i,j)*xx1(i)%Jumlah
keluaran hidden layer
        end
        yh11(j)=lmdh*zh11(j) % keluaran hidden=
Jumlah HL dikali fungsi aktivasi linear
    end
    % menghitung output neuron
    zo11=0;%z output
    for j=1:jeh11
        zo11=zo11+woh11(j)*yh11(j);
    end
    zo11=zo11+0.55;
    yn11=lmdo*zo11; % fungsi aktivasi linear
                    % yn1=NKp
MKp=yn11;
```

5. Program *Mapping Neural Network* untuk Parameter Ki

```
function MKi=mappingKi(in22)
global tt jin22 jeh22 lmdh2 lmdo2 alph2 alpo2
wih22 woh22
tt = in22(6);%clock
% proses inisialisasi

if tt==0
    jin22 = 5; %Jumlah unit input
    jeh22 = 3; %Jumlah unit HL.
    wih22 =
[0.943921552196291,0.943921552196291,
0.943921552196291;
```

```
0.944394132497019,  0.944394132497019,
0.944394132497019;
0.951587717034712,  0.951587717034712,
0.951587717034712;
0.952116868903340,  0.952116868903340,
0.952116868903340;
            0.952653103812311,
0.952653103812311,   0.952653103812311];%bobot
input hidden
    woh22 = [0.0676753751743662
             0.0676753751743662
             0.0676753751743662];%bobot output
hidden
    lmdh2=1;%fungsi aktivasi hidden layer
    lmdo2=1;%fungsi aktivasi output layer
    alph2=0.001;%learning rate hidden
    alpo2=0.001;%learning rate output
end
    for x=1:jin22
        xx(x)=in22(x);
    end
    % Perhitungan forward
    % Menghitung output layer hidden
    for j=1:jeh22
        zh22(j)=0;%z hidden
        for i=1:jin22
zh22(j)=zh22(j)+wih22(i,j)*xx(i)%Jumlah keluaran
hidden layer
        end
        yh22(j)=lmdh2*zh22(j) % keluaran hidden=
Jumlah HL dikali fungsi aktivasi linear
    end
    % menghitung output neuron
    zo22=0;%z output
    for j=1:jeh22
        zo22=zo22+woh22(j)*yh22(j);
    end
    zo22=zo22+0.05
    yn22=lmdo2*zo22; % fungsi aktivasi linear
                % yn1=NKp
```

```
MKi=yn22;
```

6. Program *Mapping Neural Network* untuk Parameter Kd

```
function MKd=mappingKd(in33)
global tt jin33 jeh33 lmdh3 lmdo3 alph3 alpo3
wih33 woh33
tt = in33(6);%clock
% proses inisialisasi
if tt==0
    jin33 = 5; %Jumlah unit input
    jeh33 = 3; %Jumlah unit HL.
    wih33 =[1.03016495778720,1.03016495778720
1.03016495778720;
1.02403146138420,    1.02403146138420,
1.02403146138420;
1.18427377830441,    1.18427377830441,
1.18427377830441;
1.18587518815143,    1.18587518815143,
1.18587518815143;
1.18747768483039,    1.18747768483039,
1.18747768483039];%bobot input hidden
    woh33 =[-0.0156742594538769
            -0.0156742594538769
            -0.0156742594538769];%bobot output
hidden
    lmdh3=1;%fungsi aktivasi hidden layer
    lmdo3=1;%fungsi aktivasi output layer
    alph3=0.001;%learning rate hidden
    alpo3=0.001;%learning rate output
end
    for x=1:jin33
        xx(x)=in33(x);
    end
    % Perhitungan forward
    % Menghitung output layer hidden
    for j=1:jeh33
        zh33(j)=0;%z hidden
        for i=1:jin33
```

```
zh33(j)=zh33(j)+wih33(i,j)*xx(i)%Jumlah keluaran
hidden layer
        end
        yh33(j)=lmdh3*zh33(j) % keluaran hidden=
Jumlah HL dikali fungsi aktivasi linear
    end
    % menghitung output neuron
    zo33=0;%z output
    for j=1:jeh33
        zo33=zo33+woh33(j)*yh33(j);
    end
    zo33=zo33+0.05
    yn33=lmdo3*zo33; % fungsi aktivasi linear
                % yn1=NKp
MKd=yn33;
```

RIWAYAT PENULIS

Uci Adriati lahir di Bogor pada tanggal 24 September 1990 tepat Hari Agraria Nasional. Penulis merupakan putri ketiga dari pasangan Ir. Suripto, M.Eng dan Sri Swidati yang bertempat tinggal di Ibukota Cibinong, Kabupaten Bogor. Semasa sekolah SD hingga SMA tahun 1996-2008 penulis aktif di organisasi Pramuka, OSIS, Paskibra, Taekwondo dan beranjak menjadi mahasiswi Teknik Listrik di Politeknik Negeri Jakarta (PNJ) tahun 2010-2013 penulis menjadi pendiri dan aktivis Divisi *Regular Practice* dalam organisasi *Polytechnic English Club* (PEC) yang kini setiap tahunnya anggota PEC bertambah lebih dari 200 mahasiswa dengan kemampuan bahasa Inggris yang baik, sehingga beberapa anggota PEC termasuk penulis berpeluang terpilih sebagai *Vacation Trainee* dan *Ambassador* di *Summer Internship Program* oleh perusahaan Schlumberger. Setelah lulus sebagai Ahli Madya Teknik Elektro PNJ, awal tahun 2014 penulis melanjutkan kuliah ke Program Lintas Jalur S1 di Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan mengambil Bidang Studi Teknik Sistem Pengaturan, Jurusan Teknik Elektro. Sejak kuliah semester 3-4, penulis berkesempatan menjadi Asisten Laboratorium Teknik Pengaturan untuk praktikum Sistem Pengaturan Digital. Latar belakang ayah yang seorang dosen di Teknik Sipil PNJ dan kedua orang kakak yang berprofesi sebagai aparatur negara di bidang hukum, menjadi motivasi bagi penulis untuk menyelesaikan masa pendidikan di ITS dengan sebaik-baiknya agar kelak dapat menjadi orang yang berguna bagi nusa, bangsa dan agama. Pada bulan Januari 2016, penulis mengikuti seminar dan ujian Tugas Akhir yang diadakan Teknik Elektro, Fakultas Teknologi Industri ITS sebagai salah satu persyaratan untuk memperoleh gelar Sarjana. Alhamdulillah atas doa, kerja keras dan kehendak Allah SWT, penulis dapat lulus menjadi Sarjana Teknik Elektro ITS.

Email: uci.adriati@gmail.com

Halaman ini sengaja dikosongkan